

Módulo 1

Proyecto CIAA: Utilizando la EDU-CIAA

Autores: Joaquín Rodríguez, Juan Pablo Vecchio

Tutor: Ing. Marcelo Pistarelli

Supervisor: Ing. José Ignacio Sosa

Asesor: Ing. Gustavo Muro

Contenido

1	Introducción.....	3
2	La Computadora Industrial Abierta Argentina (CIAA)	3
3	Instalación y configuración del entorno IDE.....	6
3.1	Instalación del IDE.....	6
3.2	Error de Seguridad de Windows	11
3.3	Fin de la instalación.....	12
3.4	OpenOCD.....	14
3.5	Corrección del Driver FTDI.....	14
3.6	Desinstalación	17
4	Configuración del entorno CIAA-IDE	17
4.1	Workspace.....	18
5	Primeros Pasos.....	18
5.1	Proyecto “Blinking”	18
5.2	Indexación de cabeceras.....	21
5.3	Configuración del Makefile	24
6	Debug con Windows	28
7	Debug en placa EDU-CIAA y Entorno IDE	34
7.1	Configuración del entorno CIAA-IDE.....	34
7.2	Compilación del proyecto	36
7.3	Depuración sobre la placa: configuración de OpenOCD para Debug	36
7.4	Posible problema: “No reconocimiento”.....	39

1 Introducción

El siguiente documento pretende explicar todos los pasos necesarios para comenzar a desarrollar proyectos sobre la plataforma EDU-CIAA. Comenzaremos con todo lo referente a la instalación del entorno, la configuración del mismo para poder cargar el Firmware desarrollado para el proyecto, la carga de un ejemplo (llamado “Blinking”, es decir, titilación de LEDS), y la descripción de pautas para desarrollar software desde cero. Parte de esta contribución utiliza la documentación existente en la Wiki del proyecto, pero se trata de realizar una explicación más profunda, de más bajo nivel, para aquéllos que no estén tan familiarizados con la programación Open-Source.

Por otra parte, este documento está exclusivamente desarrollado para la plataforma Windows.

También se encuentra disponible un módulo exclusivamente hecho para Linux/Ubuntu, llamado ***Módulo 2: Proyecto CIAA: Primeros pasos en Ubuntu / Linux.***

2 La Computadora Industrial Abierta Argentina (CIAA)

El ***Proyecto CIAA*** nació en el año 2013 como una iniciativa conjunta entre el sector académico y el industrial, representados por la ***ACSE*** y ***CADIEEL***, respectivamente.

Los objetivos del ***Proyecto CIAA*** son:

1. Impulsar el desarrollo tecnológico nacional.
2. Darle visibilidad positiva a la electrónica argentina.
3. Generar cambios estructurales en la forma en la que se desarrollan y utilizan los conocimientos.

Todo esto en un marco de trabajo ***libre, colaborativo y articulado*** entre industria y academia.

Para lograr los objetivos, el primer paso fue articular el trabajo de decenas de Instituciones, Universidades, Empresas y Desarrolladores para diseñar la primera versión de la CIAA, denominada “***CIAA-NXP***” por estar basada en un procesador de la empresa ***NXP Semiconductors***. Las Figura 1 y Figura 2 muestran el aspecto de esta placa, con indicaciones sintéticas de sus prestaciones.

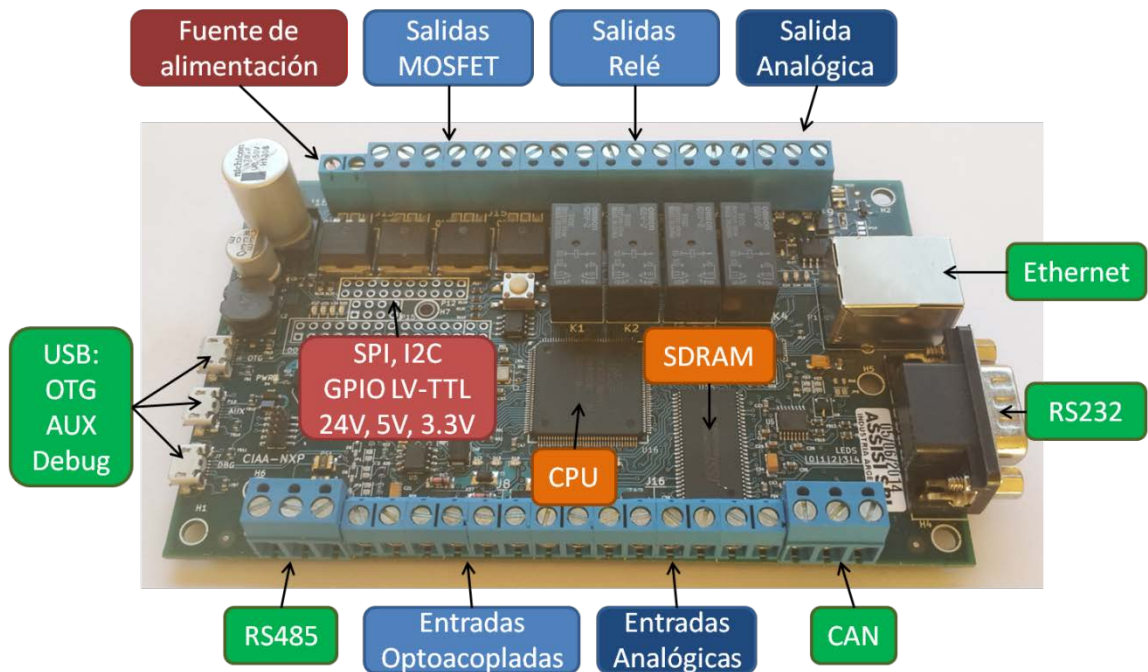


Figura 1: Vista frontal de la placa CIAA y sus principales componentes

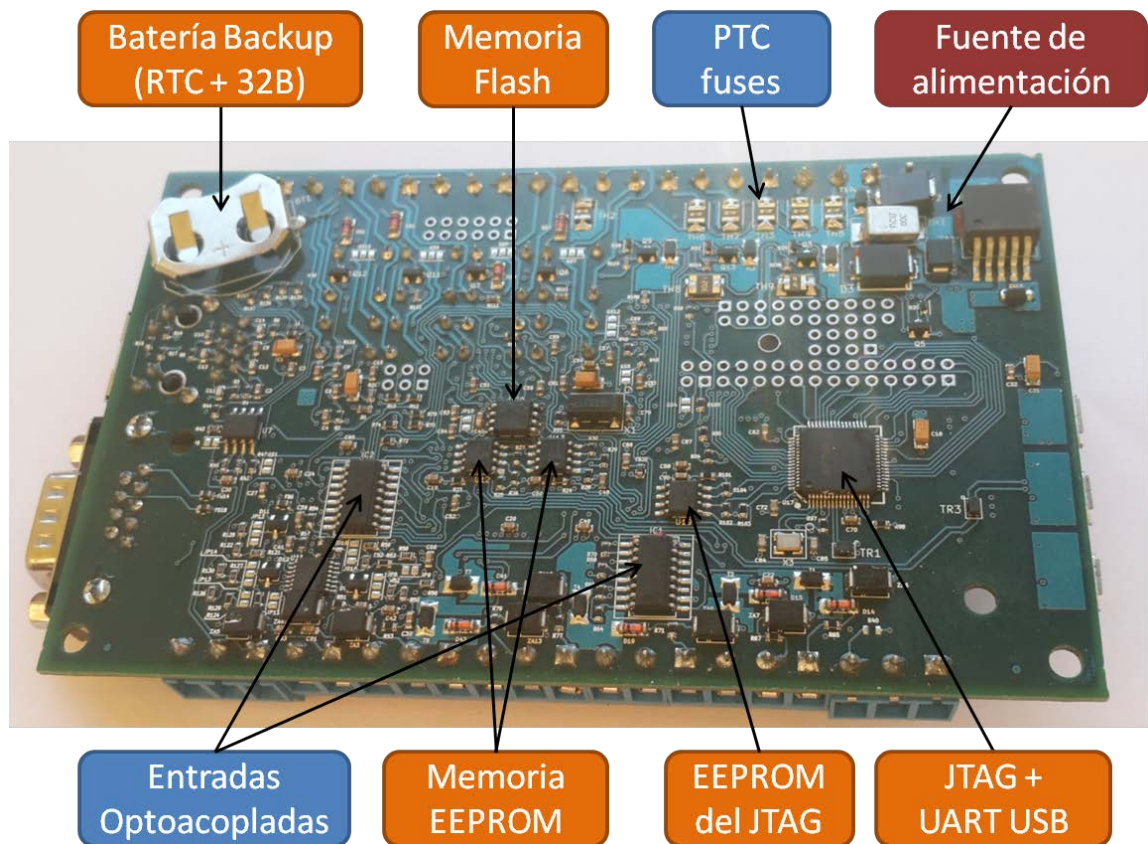


Figura 2: Vista posterior de la placa CIAA y sus principales componentes

La **CIAA-NXP** es la primera y única computadora del mundo que reúne dos cualidades:

1. Es **industrial**, ya que su diseño está preparado para las exigencias de confiabilidad, temperatura, vibraciones, ruido electromagnético, tensiones, cortocircuitos, etc., que demandan los productos y procesos industriales.
2. Es **abierta**, ya que toda la información sobre su diseño de hardware, firmware, software, etc. está libremente disponible en internet bajo la Licencia BSD, para que cualquiera la utilice como quiera.

Para avanzar aún más se desarrollaron versiones de la CIAA basadas en procesadores de otras marcas, como la **CIAA-FSL**, la **CIAA-INTEL**, la **CIAA-PIC**, etc.

En consecuencia, la CIAA además de ser la primera computadora industrial abierta, es también **la primera computadora realmente libre del mundo**, ya que su diseño no está atado a los procesadores de una determinada compañía, como ocurre con otras computadoras abiertas.

No hay que perder de vista que el **Proyecto CIAA** es mucho más que hardware, ya que también incluye un **entorno IDE** para su programación en **lenguaje C**, el **soporte de Linux**, un entorno de programación en **lenguaje tipo PLC**, el diseño de un gabinete y los primeros diseños de algunos de sus circuitos integrados.

Además, se diseñó una versión educativa de la plataforma, la **EDU-CIAA**, más simple y de menor costo, para lograr un impacto en la **enseñanza primaria, secundaria y universitaria**. La Figura 3 muestra una imagen de la misma.

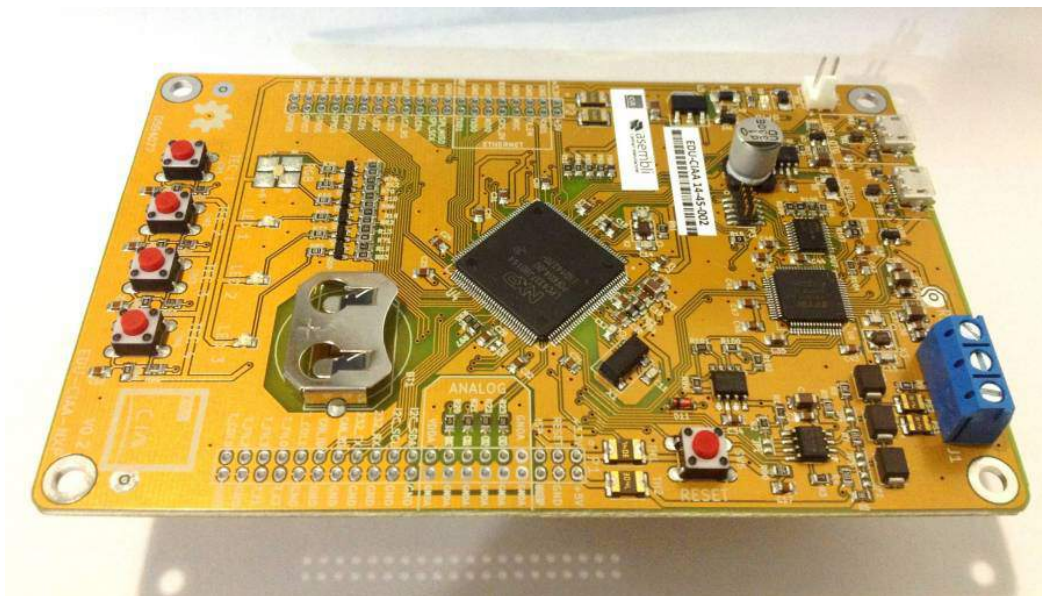


Figura 3: Versión educativa de la Computadora Industrial (EDU-CIAA)

Las distintas versiones de la EDU-CIAA utilizan los mismos procesadores y programas que la CIAA, de modo que los estudiantes aprenden a resolver problemas reales, y pueden aplicar sus conocimientos y desarrollos en aplicaciones laborales y emprendimientos tecnológicos. Otro

detalle a tener en cuenta es que la EDU-CIAA es una versión simplificada de la CIAA. La misma cuenta con el mismo controlador y el mismo circuito integrado encargado de la depuración sobre la placa. Esto es lo mínimo que se necesita para conectar la placa a una computadora y programar sobre ella. Adicionalmente, cuenta con LEDs y pulsadores que pueden usarse para hacer algunas pruebas sencillas. El resto de los componentes se tendrán que agregar en forma de módulos, enchufados a los conectores ubicados en los laterales de la misma., lo que se justifica porque la EDU-CIAA podría usarse como una placa para hacer los primeros ensayos del código desarrollado, y luego trasladar todo el proyecto directamente a una placa CIAA, y realizar las pruebas finales con las interfaces industriales que ésta posee.

3 Instalación y configuración del entorno IDE

3.1 Instalación del IDE

El **IDE** (*inglés. Integrated Development Environment = Entorno de Desarrollo Integrado*) provee al desarrollador de Firmware (programador en C/C++) la posibilidad de trabajar en un ambiente amigable y *plug & play*.

Todas las herramientas necesarias para poder desarrollar aplicaciones en el Firmware colaborativo creado para el proyecto se instalarán en forma automática: al descargar el instalador, se provee una copia del Firmware y de los ejemplos, en su última versión. El paquete de instalación incluye:

- **Eclipse**: es el entorno base utilizado para desarrollo de aplicaciones, es decir, el **software IDE**.
- **PHP (Hypertext Pre-processor)** es un lenguaje de programación de uso general de código desde el lado del servidor, originalmente diseñado para el desarrollo de contenido dinámico. En este caso, se utiliza solamente en forma de *scripts* para poder generar algunos archivos del **Sistema Operativo OSEK**.
- **Cygwin**: es una consola que se ejecuta en Windows, de modo de emular la consola de comandos de Linux. Cuenta con todos los comandos, y el **compilador GCC**, propio del sistema operativo libre.

Para descargar el instalador, hay que dirigirse a la página oficial del proyecto, siguiendo este [link](#). En ella, se encontrará el software IDE Suite. El mismo es compatible con las versiones de Windows XP, Windows Vista, Windows 7 y Windows 8, tanto en sus versiones de 32 como de 64 bits. Junto con la descarga, se incluye una copia del Firmware.

Cualquier inquietud, o inconveniente que surja respecto al software, puede dirigirse a la lista de mails: ciaa-ide@googlegroups.com.

El instalador provee todo el entorno necesario, con el cual se podrán instalar y configurar automáticamente de forma sencilla la gran mayoría de las herramientas necesarias para trabajar con la CIAA. Su uso es sumamente intuitivo: se recomienda no cambiar el directorio de instalación y, de ser necesario, elegir un nombre que NO contenga espacios. En las Figura 4 y Figura 5 se pueden ver las primeras dos ventanas correspondientes al instalador.



Figura 4: Arranque del instalador del software IDE

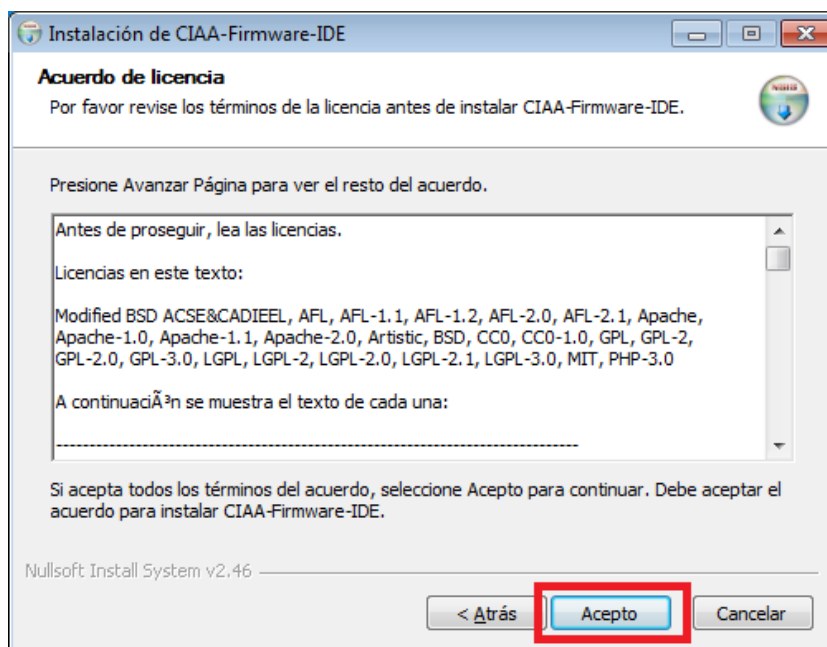
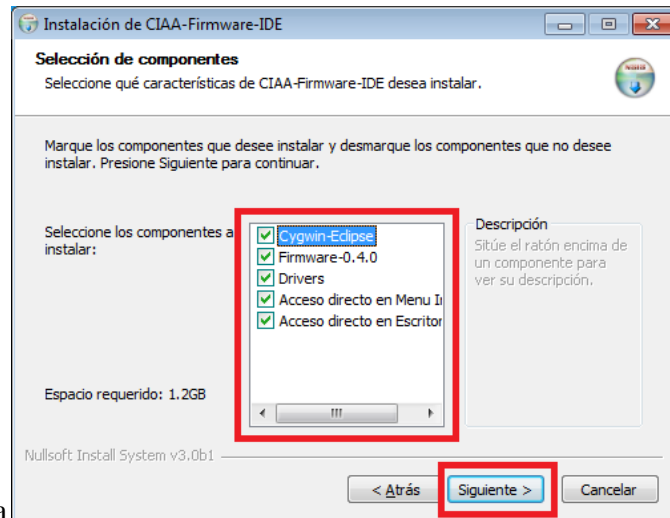


Figura 5: Acuerdo de licencia del instalador



En la ventana siguiente, mostrada en la

Figura 6, se deben elegir los componentes que se desea instalar. Si usted no posee una EDU-CIAA, no es necesario que instale los drivers: si la adquiere en algún momento posterior, dado que los drivers se instalan junto con el IDE, los mismos quedarán en la carpeta de destino para su instalación en forma manual. Otra manera de instalar los controladores es ejecutar el instalador del CIAA-IDE Suite y tildar únicamente la opción **drivers** al momento de seleccionar los componentes a instalar.

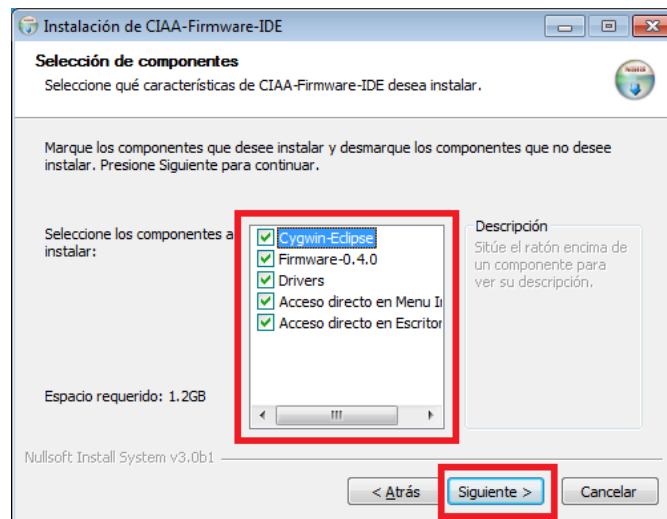


Figura 6: Selección de componentes del instalador

A continuación deberá colocar la dirección (*path*) en donde desea instalar el entorno. La ventana correspondiente se muestra en la Figura 7. Como se indicó anteriormente, si se desea cambiarlo debe tenerse la precaución de no elegir una dirección donde los directorios posean espacios en sus nombres. Recomendamos no cambiar la unidad de instalación, pues en los siguientes pasos del documento se utilizarán direcciones que harán referencia a esta carpeta de instalación, y si se cambia, se deberán cambiar consecuentemente dichas direcciones

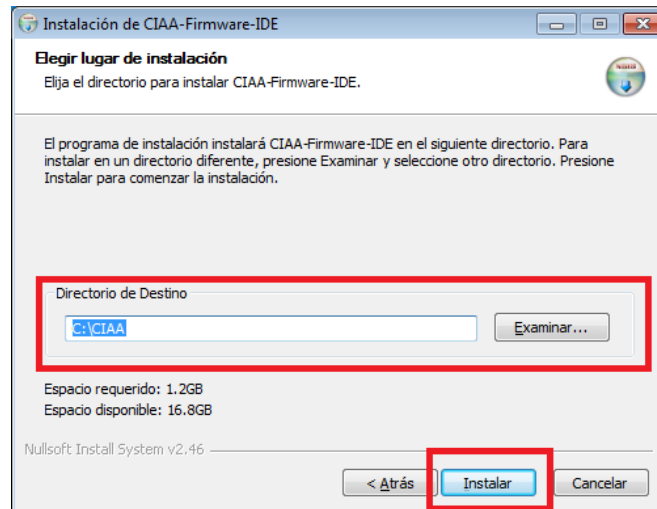


Figura 7: Elección de la ruta de instalación

Hecho lo anterior, la instalación empieza automáticamente. En un momento aparecerá una ventana emergente, similar a la que se muestra en la Figura 8, en donde el programa pregunta si disponemos del hardware, pues para la instalación del driver es necesario conectar la placa. De no ser así, aún puede continuar la instalación haciendo click en '**No**'. Por el contrario, si disponemos de la EDU-CIAA, hacemos click en '**Yes**', y emergerá otra ventana, como se muestra en la Figura 9.

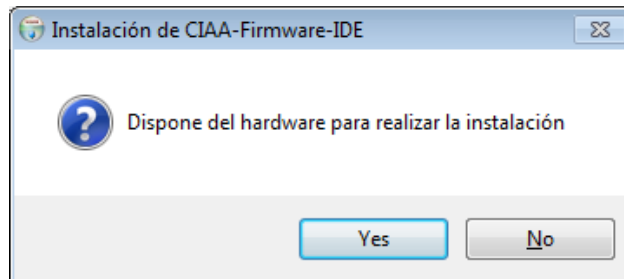


Figura 8: Instalación de los drivers: primera instancia

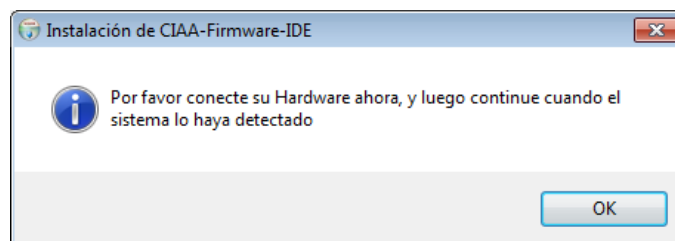


Figura 9: Instalación de drivers si se dispone del hardware

Concluida esta etapa, se procederá a la instalación de los drivers por defecto del fabricante FTDI para puerto virtual. Las imágenes correspondientes se muestran en las Figura 10, 11, 12 y 13.

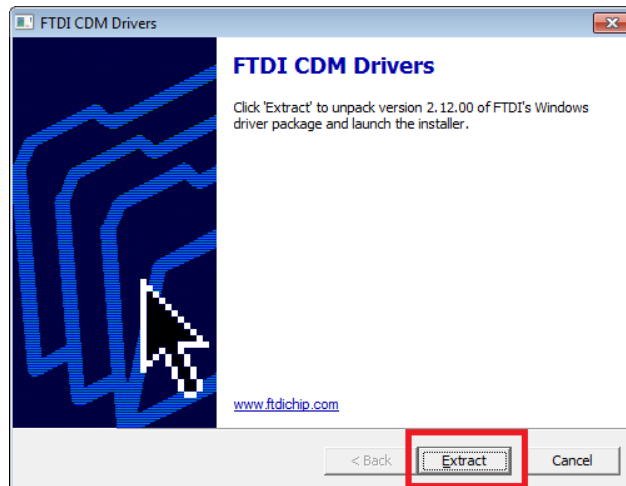


Figura 10: Instalador de drivers FTDI

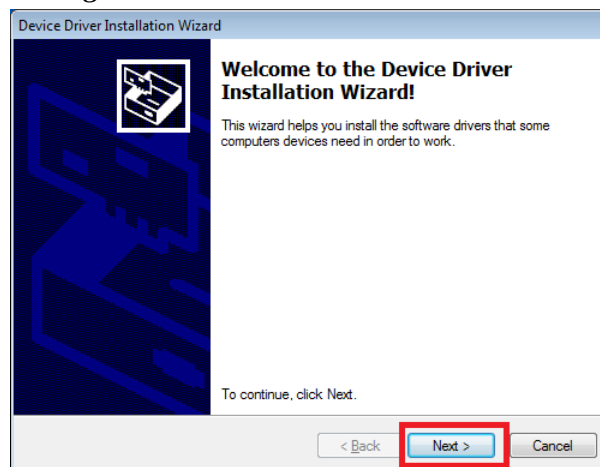


Figura 11: Instalación de drivers FTDI (cont.)

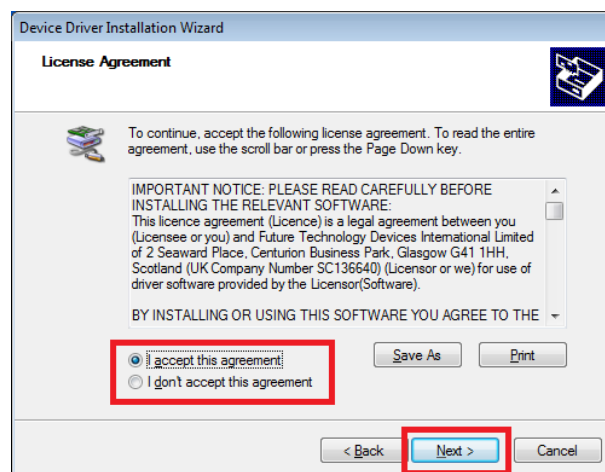


Figura 12: Instalación de drivers FTDI (cont.)

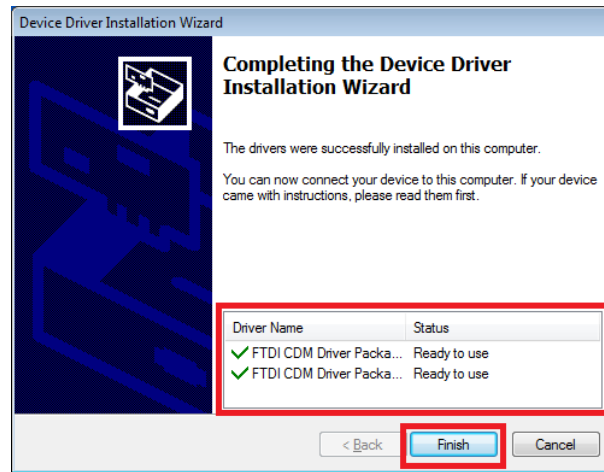


Figura 13: Instalación de drivers FTDI (fin)

El **Proyecto CIAA** está en continua expansión, y día a día se van realizando mejoras. Esto acarrea problemas, algunos de ellos (encontrados y solucionados) relacionados con los drivers: se trata de una falla en la comunicación a través del puerto virtual FTDI, que impide la correcta comunicación entre la placa y el entorno IDE. Su corrección debe efectuarse manualmente, fuera del instalador, y es posible la aparición de una ventana de error emergente como la que se muestra en la Figura 14.

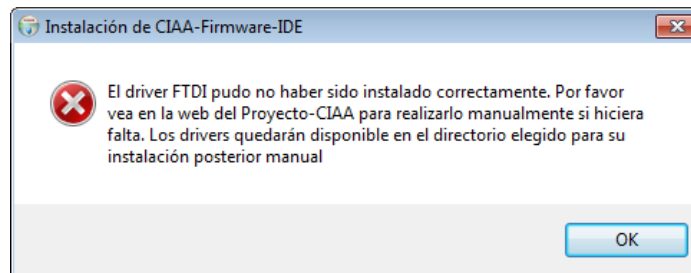


Figura 14: Aviso de falla en la instalación de los drivers

Para la corrección del driver, consulte la sección “**Corrección del Driver FTDI**” cuando termine la instalación del Software IDE.

3.2 Error de Seguridad de Windows

Bajo algunos sistemas operativos, si se detecta que el driver no posee una firma válida, muestra un error Windows Security, como muestra en la Figura 15.

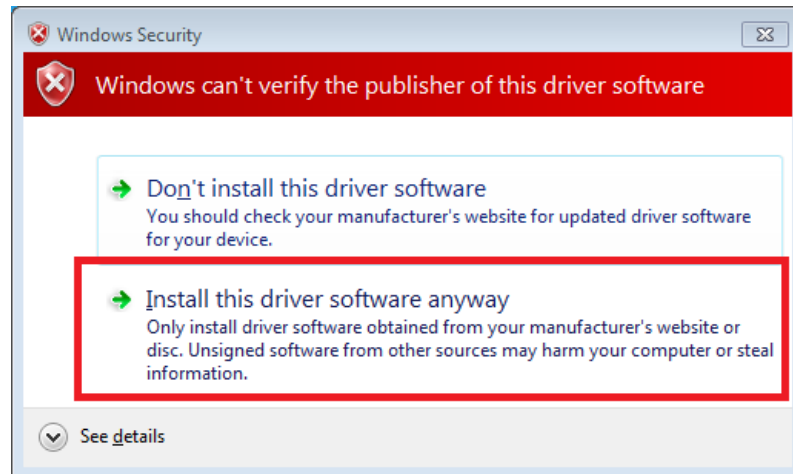


Figura 15: Windows Security Error

En este caso usted puede confiar en la procedencia y hacer clic en “*Install this driver software anyway*”.

3.3 Fin de la instalación

Las últimas ventanas del instalador se muestran en las Figura 16 y 17:

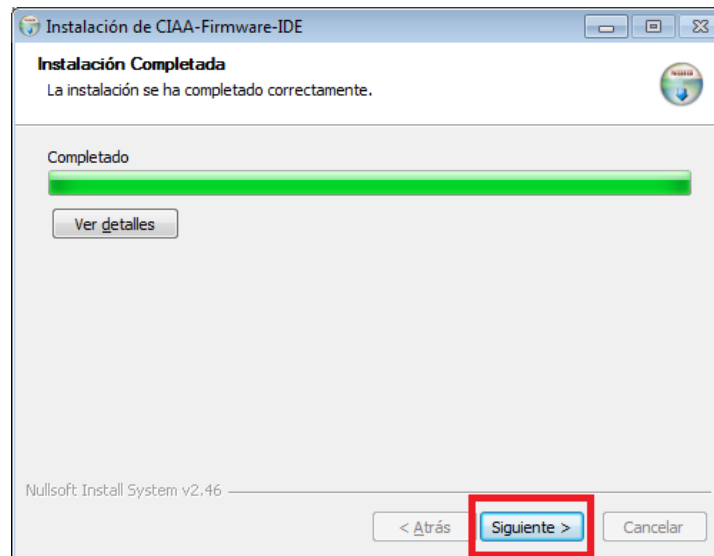


Figura 16: Fin de la copia de archivos y de la instalación de drivers

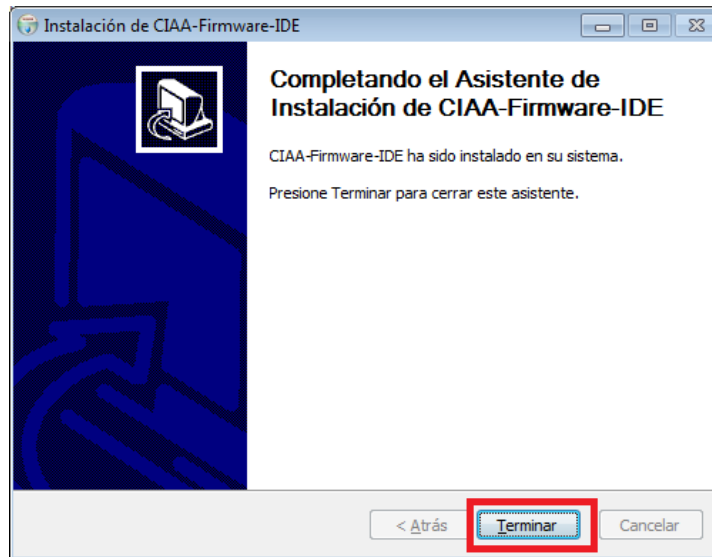


Figura 17: Fin de instalación del software IDE

Si no ocurrieron errores durante la instalación, usted dispondrá de un link en el escritorio que le permitirá abrir el CIAA-IDE y comenzar a trabajar con el Firmware.

3.4 OpenOCD

El hardware de la CIAA cuenta con un puerto USB para poder realizar la programación y depuración del programa en el microcontrolador: esto está implementado en el chip FTDI **FT2232H**.

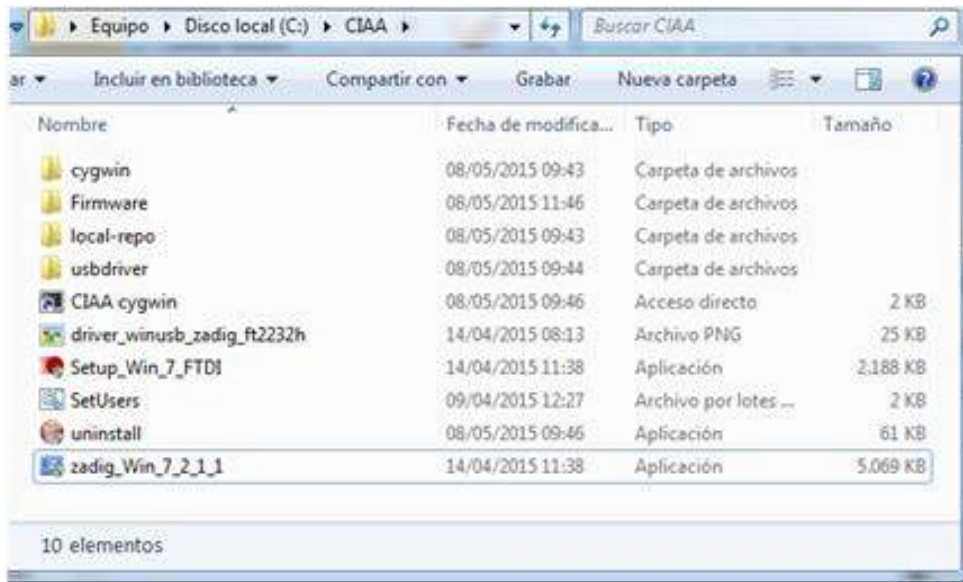
La herramienta de código abierto **OpenOCD** (*On-Chip Debugger*) es la encargada de manejar el chip FT2232H a través del USB y a la vez todo lo referido al JTAG. Con esto, el *debugger* (GDB) utilizado en el IDE-Eclipse puede hacer su tarea simplemente conectándose al puerto 3333 (TCP) que el OpenOCD mantiene en escucha esperando la conexión.

Téngase en cuenta que el chip FT2232H posee 2 canales de comunicación independientes (A y B), pero ambos salen por el mismo puerto USB. Es decir, la computadora a la que está conectado verá 2 dispositivos distintos (en realidad uno compuesto): uno de estos dispositivos será el que conecta al JTAG manejado por OpenOCD (como se mencionó), y el otro se verá como un puerto virtual COM: este último puede servir principalmente para *debug*. Cada uno de estos dos dispositivos tendrá un driver asociado.

Lo primero es instalar los drivers por defecto del fabricante FTDI para puerto virtual (VCP). En el Administrador de Dispositivos deberían aparecer 2 nuevos puertos COM, tal como se muestra en la Figura 18.

3.5 Corrección del Driver FTDI

Un inconveniente que se nos puede presentar al momento de conectar nuestra placa con el sistema operativo, está vinculado con los drivers de la placa EDU-CIAA que se incluyen dentro del instalador descargado desde la página. El instalador incluye en la carpeta donde se instaló el software (Por defecto, **C:\CIAA**) un programa que configura el driver del controlador serie, emulado por la placa, para que uno de ellos pueda ser utilizado como interfaz JTAG. Dicho programa se llama **Zadig_Win_7_2_1_1.exe**.



Las

Figura 19 y Figura 20 muestran, respectivamente, el explorador de Windows situado en la carpeta del software y el entorno del programa corrector.

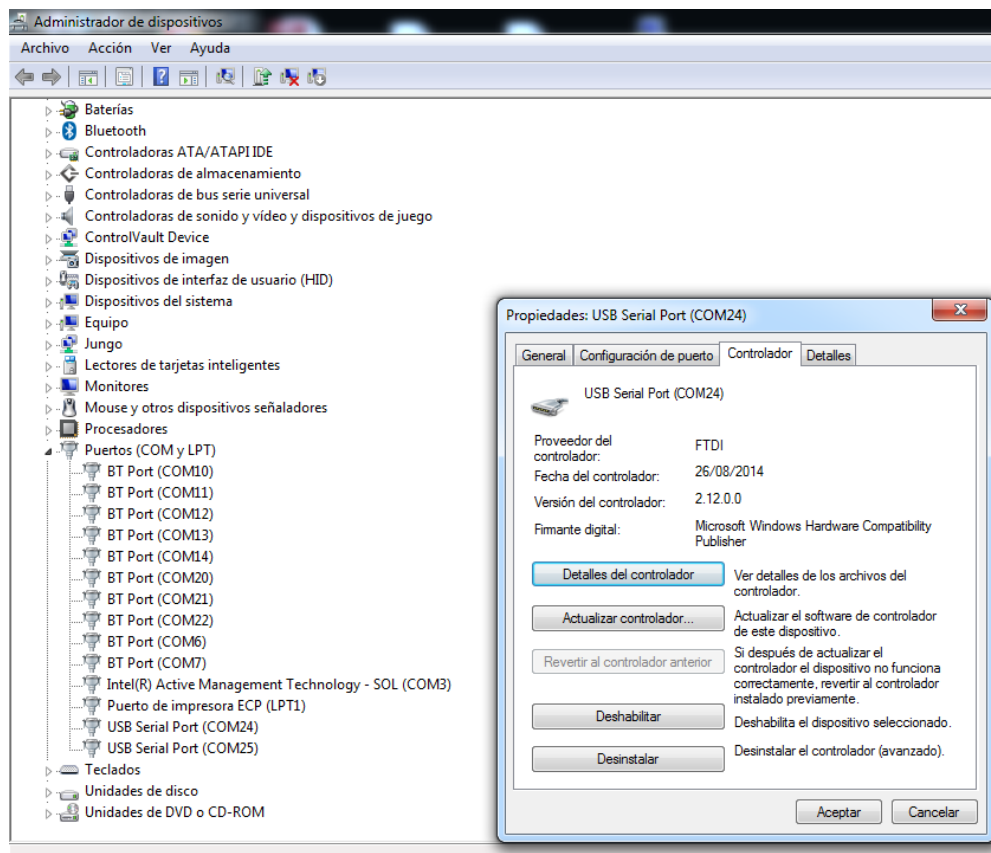


Figura 18: Administrador de dispositivos con la EDU-CIAA conectada

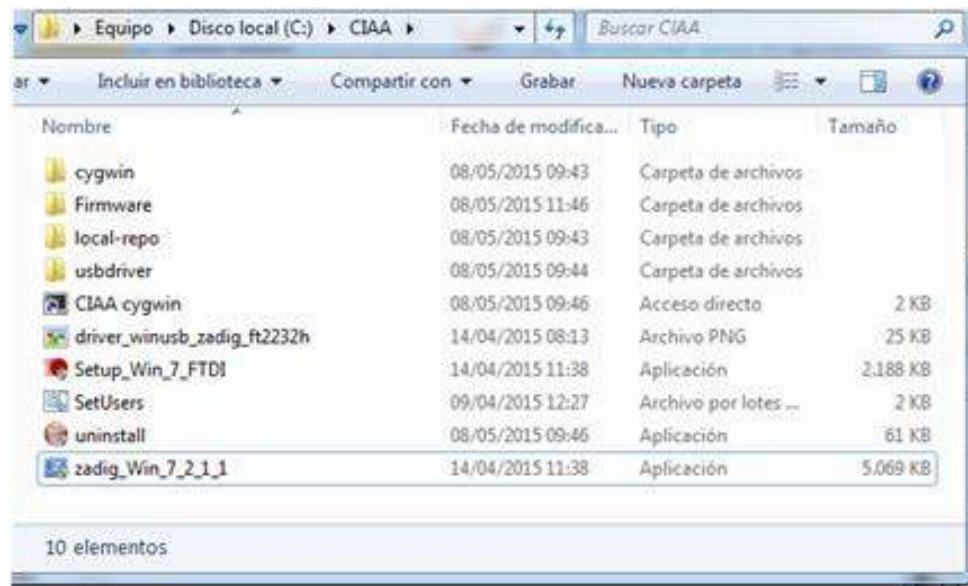


Figura 19: Ventana del Explorador de Windows en la carpeta del Software IDE

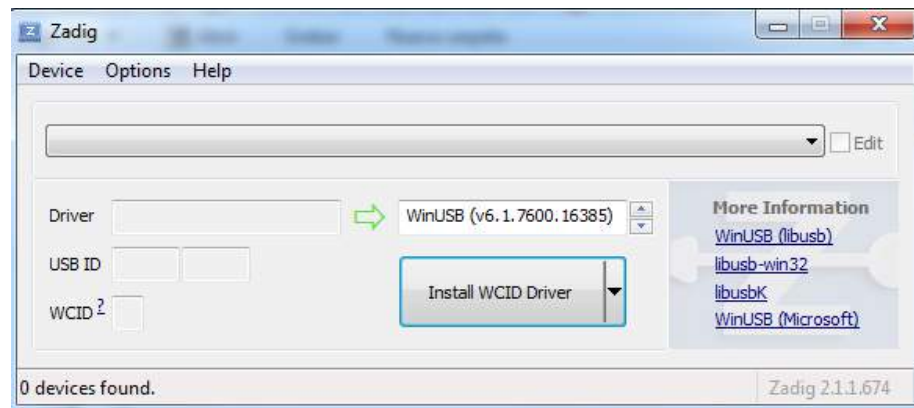


Figura 20: Entorno del software corrector Zadig para Windows

Para corregir los drivers, conectamos la placa a través del cable USB, hacemos click en **Options** – > **List All Devices**. Aparecerá una lista de dispositivos de comunicación relacionados al USB. Tenemos que buscar aquellos cuyos nombres tengan relación con el puerto serie (puede aparecer Dual RS232-HS, USB Serial Converter, o algo similar). Por lo general, aparecerán 2 con el mismo nombre, excepto que uno es **Interface 0** y el otro **Interface 1**, como se muestra en la Figura 21 (la lista de drivers que se muestra puede diferir, dependiendo de la computadora que se utilice).

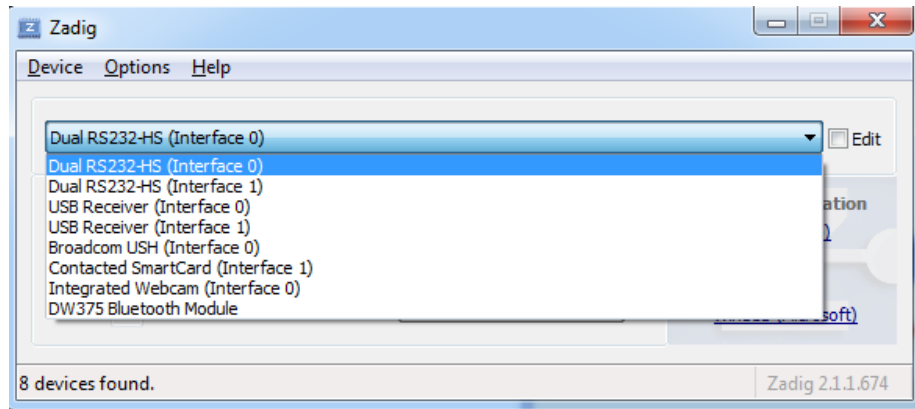


Figura 21: Lista de dispositivos vinculados a USB

Para configurar el driver:

- a) seleccionar la **Interfase 0**
- b) elegir el “**WinUSB v6.1**”
- c) hacer click en el botón “**Replace Driver**”.

La ventana ya configurada se muestra en la Figura 22.

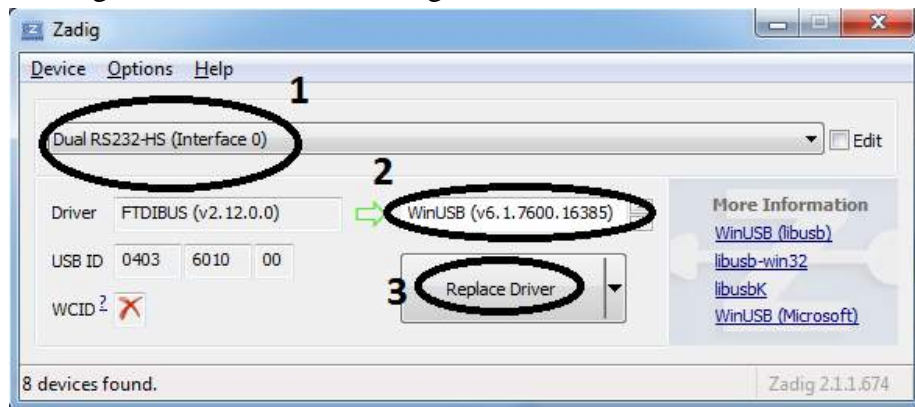


Figura 22: Configuración del Zadig para el reemplazo del driver

3.6 Desinstalación

Si se instaló el Software de CIAA-IDE y luego se desea desinstalarlo, se debe tener especial cuidado en quitar cualquier contenido que se quiera conservar de la carpeta C:\CIAA, o el directorio de instalación elegido. Esto se debe a que el desinstalador del Software CIAA-IDE elimina el directorio y todo su contenido.

4 Configuración del entorno CIAA-IDE

4.1 Workspace

Al iniciar el entorno CIAA-IDE, éste solicita que seleccione una carpeta de trabajo, como se muestra en la Figura 23:

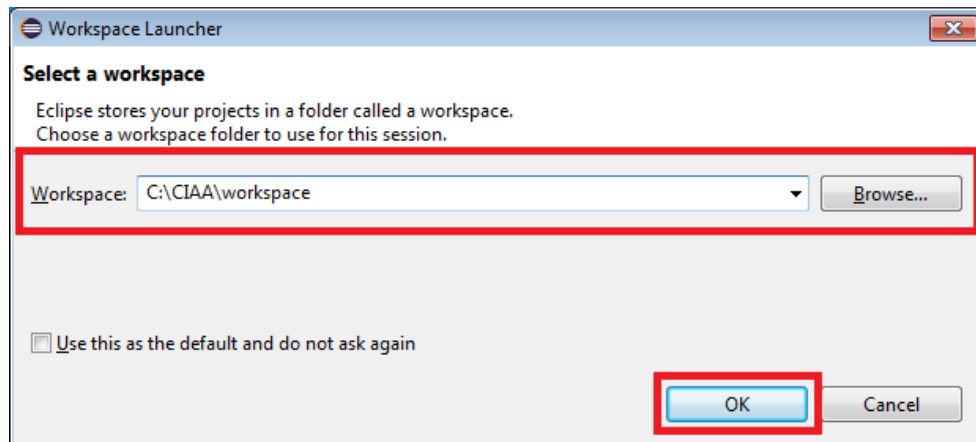


Figura 23: Ventana inicial para ubicar el espacio de trabajo (Workspace)

Todo lo relativo al proyecto en que se esté trabajando (configuraciones, variables globales del entorno y demás) se guarda en esta carpeta. Si siempre se va a utilizar la misma ubicación, se puede tildar la opción “*Use this as the default and do not ask again*”. En cualquier otro caso, cada vez que se abra el entorno, se preguntará por esta ubicación. En particular puede elegirse dentro del directorio de instalación del CIAA-IDE sin ningún problema.

5 Primeros Pasos

5.1 Proyecto “Blinking”

Para el usuario sin experiencia con esta plataforma de desarrollo, se aconseja abrir el proyecto del Firmware, para luego compilar y ejecutar un proyecto de ejemplo, llamado **Blinking**. Para ello debemos ir al menú '*File→New→Makefile Project with Existing Code*', como se muestra en la Figura 24.

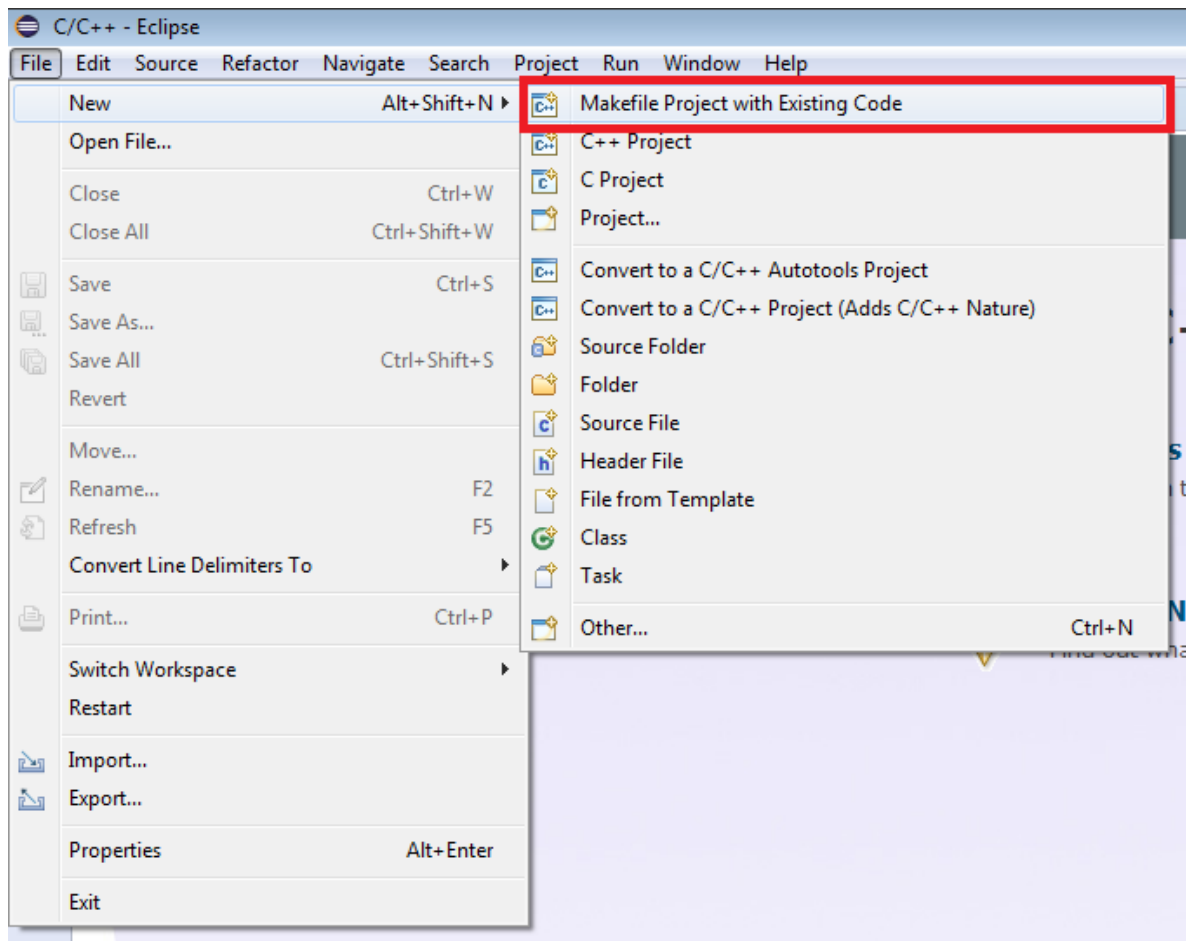


Figura 24: Comienzo de la carga del código existente

En el caso de no haber instalado el Firmware junto al entorno CIAA-IDE, hay 2 opciones:

- se puede descargar desde la web, siguiendo las instrucciones en la sección Código Fuente (utilizando el directorio de repositorios, cuyo enlace se encuentra [aquí](#)), o bien
- se puede volver a ejecutar el Instalador del Software-IDE de la CIAA y, en la ventana de componentes, seleccionar únicamente el ítem Firmware.

Seguidamente aparecerá una ventana (Figura 25) en donde se debe indicar el proyecto que se va a cargar: en este caso elegiremos la carpeta Firmware que, si usamos los directorios por defecto, se encuentra en la ubicación:

C:\CIAA\Firmware

En la ventana ***Toolchain for Indexer Settings*** seleccionaremos la opción ***<none>***, lo cual dejará las opciones por defecto, configuradas en el Makefile.

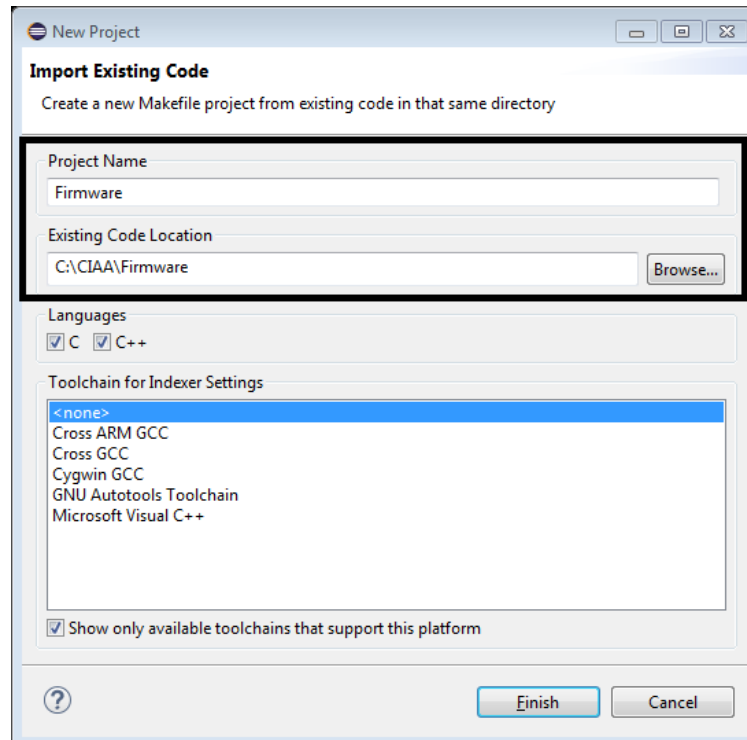


Figura 25: Creación de un nuevo proyecto usando un código existente

Una vez creado el proyecto, cerramos la pestaña **Welcome** con la que inicia Eclipse, y nos encontraremos con un entorno como el que se presenta en la Figura 26.

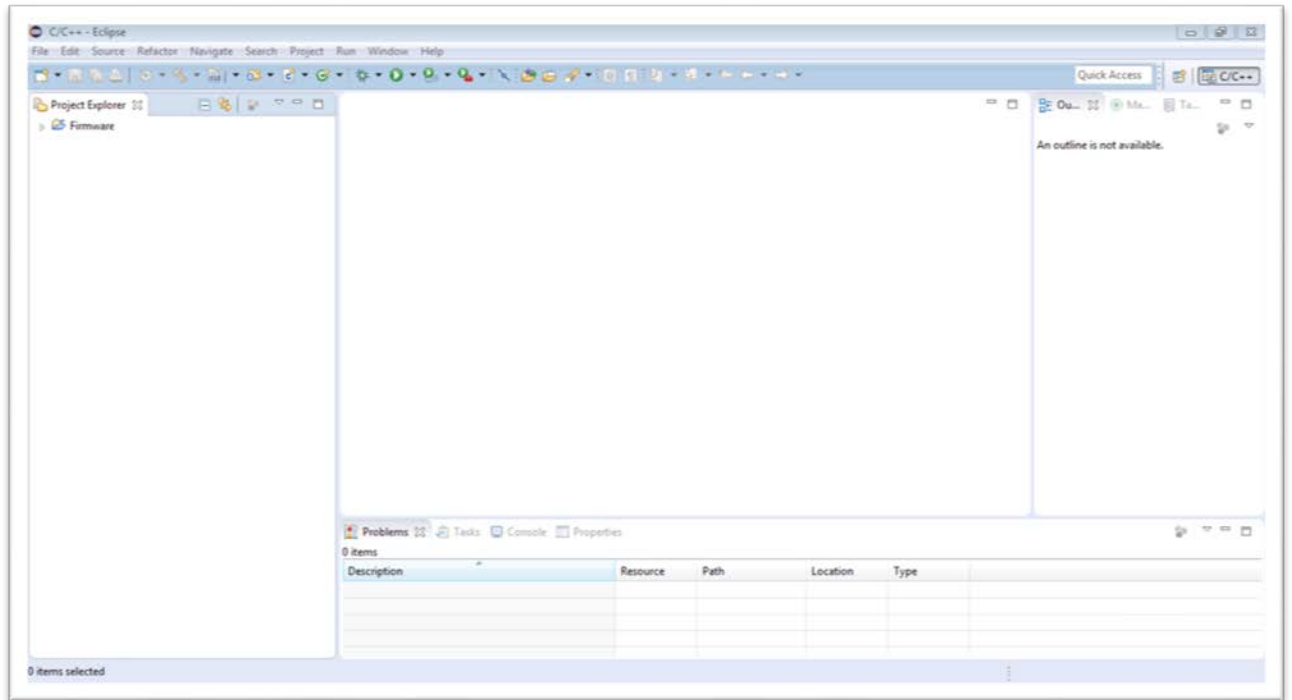


Figura 26: Entorno IDE-CIAA con el Firmware cargado

5.2 Indexación de cabeceras

Para la compilación del proyecto es necesario que el software IDE sepa dónde encontrar las cabeceras del estándar **POSIX**.

POSIX (*Portable Operating System Interface*) es un conjunto de interfaces estándar para sistemas operativos basadas en **UNIX**. Esta estandarización fue necesaria para que los distintos fabricantes de computadoras o desarrolladores de programas pudieran desarrollar sus aplicaciones independientemente de la plataforma en la cual iba a correr.

Para poder indexar esas definiciones se deben agregar los archivos **Includes** del **GCC** (*GNU Compiler Collection*). Esto se efectúa sobre la pestaña **Path and Symbols**→**Includes**, en una ventana similar a la que muestra la Figura 27.

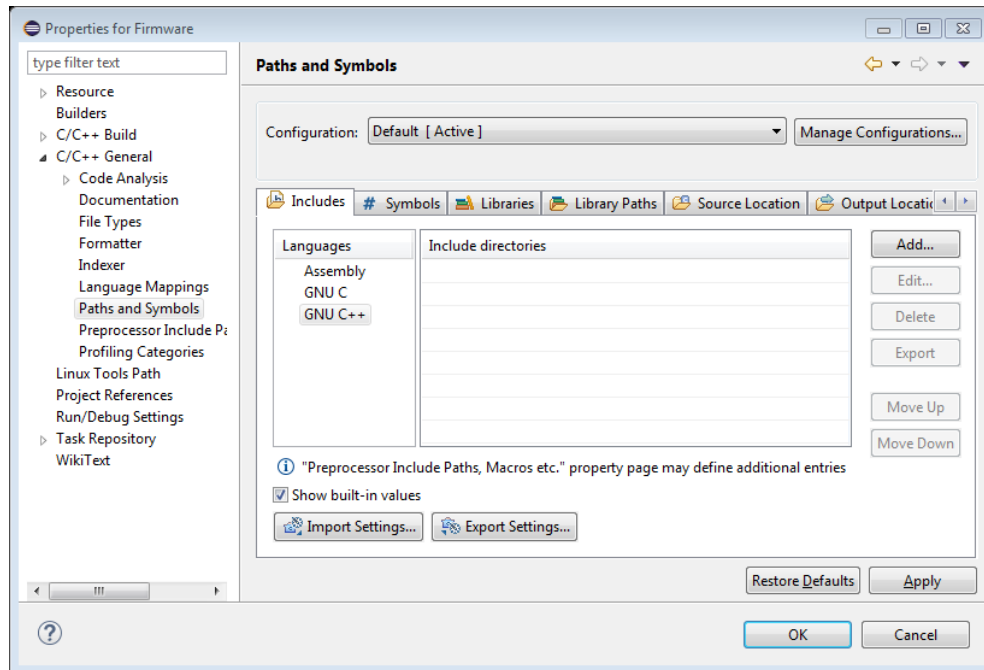


Figura 27: Indexación de las definiciones POSIX

En esta ventana seleccionar '**GNU C**' ó '**GNU C++**' según las POSIX que queramos agregar, y luego presionar el botón '**Add**'. Emergerá la ventana mostrada en la Figura 28, en la cual se debe hacer click en el botón '**File System...**' y luego buscar la carpeta correspondiente a agregar.

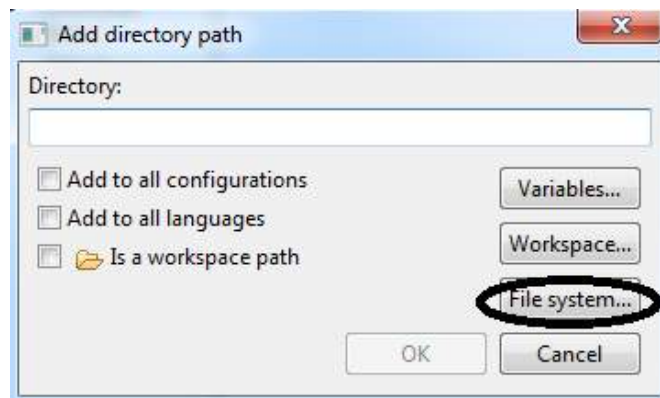


Figura 28: Indexar POSIX

Los ***Includes*** que deben configurarse, según el lenguaje (*Language*) elegido, son los siguientes:

Language = 'GNU C'

- `<dIDE>\cygwin\usr\include`. Si se aplican los directorios por defecto, éste quedaría `C:\CIAA\cygwin\usr\include`

- <dIDE>\cygwin\lib\gcc\i686-pc-cygwin\4.9.2\include
(por defecto: C:\CIAA\cygwin\lib\gcc\i686-pc-cygwin\4.9.2\include)
- (dIDE = directorio de instalación del software-IDE: por defecto es C:\CIAA)

Language = 'GNU C++'

- <dIDE>\cygwin\usr\include
(por defecto: C:\CIAA\cygwin\usr\include)
 - <dIDE>\cygwin\lib\gcc\i686-pc-cygwin\4.9.2\include\c++
(por defecto: C:\CIAA\cygwin\lib\gcc\i686-pc-cygwin\4.9.2\include\c++)
- (dIDE = directorio de instalación del software-IDE: por defecto es C:\CIAA)

Finalizada la selección, los ***Includes*** quedarán de forma similar a la mostrada en la Figura 29.

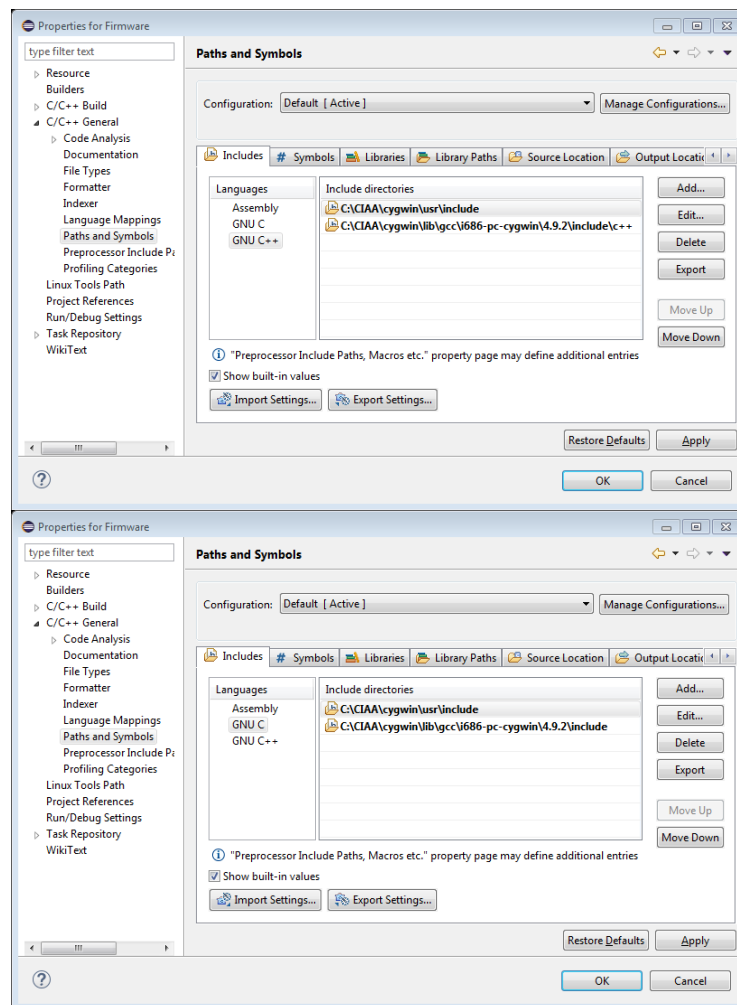


Figura 29: Fin de la configuración de Includes

5.3 Configuración del Makefile

Los archivos **Makefile** son archivos de texto escritos con una sintaxis predeterminada. Junto con la utilidad '**Make**', permiten construir el software desde sus archivos-fuente, en el sentido de organizar el código, su compilación y enlace (*link*) correcto.

El **Proyecto CIAA** tiene su propio Makefile, por lo que se debe indicarle al IDE cómo manejarse con él: de lo contrario generaría un Makefile automáticamente, lo cual nos traería muchos dolores de cabeza.

Cada vez que hacemos un comando clean, estamos borrando los archivos objeto generados previamente, pero antes de poder volver a compilar, necesitamos que esté previamente procesado el código PHP correspondiente al sistema operativo **RTOS-OSEK** (ver nota siguiente). Éste procesado se hace con un comando llamado **generate**. Para no tener que hacer cada función por separado, lo que se hace es pasarle al MakeFile el comando '**clean_generate**', que realiza ambas operaciones, en forma consecutiva y automática. Si se trabaja sin RTOS-OSEK sólo hace falta usar el comando '**clean**'.

La primera vez que se compila el proyecto, es necesario hacer un **Clean Project**. Esto ejecutará el comando **Clean_generate** del make, creando todos los archivos necesarios para la compilación con el RTOS.

NOTA

OSEK(alemán: *Offene Systeme und deren Schnittstellen für die Elektronik in Kraftfahrzeugen*) (inglés: *Open Systems and their Interfaces for the Electronics in Motor Vehicles*) es un gremio de estandarización que ha producido especificaciones para sistemas operativos embebidos, un stack de comunicación, y un protocolo de control de red para un sistema embebido automotriz, además de otras especificaciones relacionadas. OSEK (*"Sistemas abiertos y sus interfaces para la electrónica en automóviles"*) se diseñó para proveer una arquitectura de software abierta estándar para varias unidades de control electrónicas (ECU=Electronic Control Unit) incorporadas en vehículos, para facilitar la integración y portabilidad de software de diferentes proveedores, ahorrando costos y tiempos de desarrollo. Por otra parte, es un sistema operativo muy seguro, dado que todas sus funciones son determinísticas: al momento de iniciar el SO, cada tarea ya tiene asignado su espacio de memoria, evitando la necesidad de contar con instrucciones tipo '**malloc**' para la asignación dinámica de memoria. Así, ante una situación de riesgo (p.ej. al producirse un choque), no se producen retardos en la ejecución de las tareas.

Para realizar esta configuración, ubíquese en la ventana de propiedades del proyecto **Firmware**, y seleccione la rama **C/C++ Build**: se verá una pantalla similar a la que muestra la Figura 30.

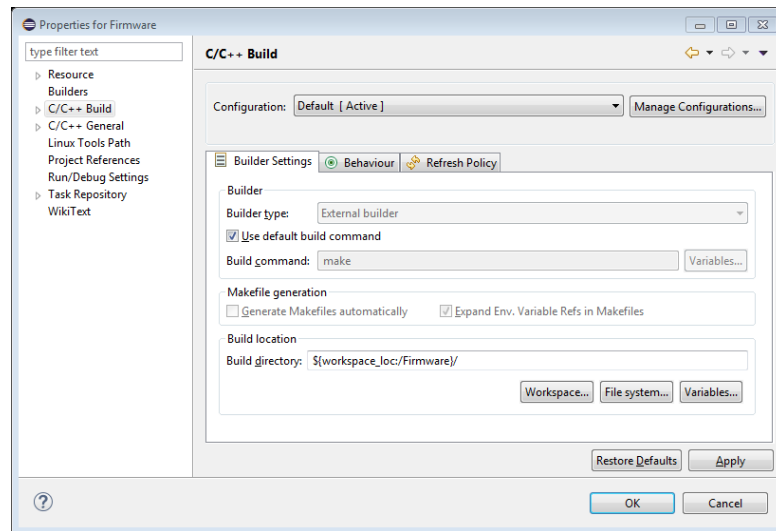


Figura 30: Configuración del Makefile

Dentro de la rama '**C/C++ Build**', configure la pestaña '**Behaviour**' como muestra la Figura 31. Las configuraciones importantes son las siguientes:

- tilde la opción '**Stop on first build error**' y destilde '**Enable parallel build**'
- destilde el casillero '**Build on resource save**' y tilde '**Build (Incremental Build)**' y '**Clean**'. En el campo **Clean**, escriba: **clean_generate**
- borre el contenido del campo **Build** y déjelo en blanco

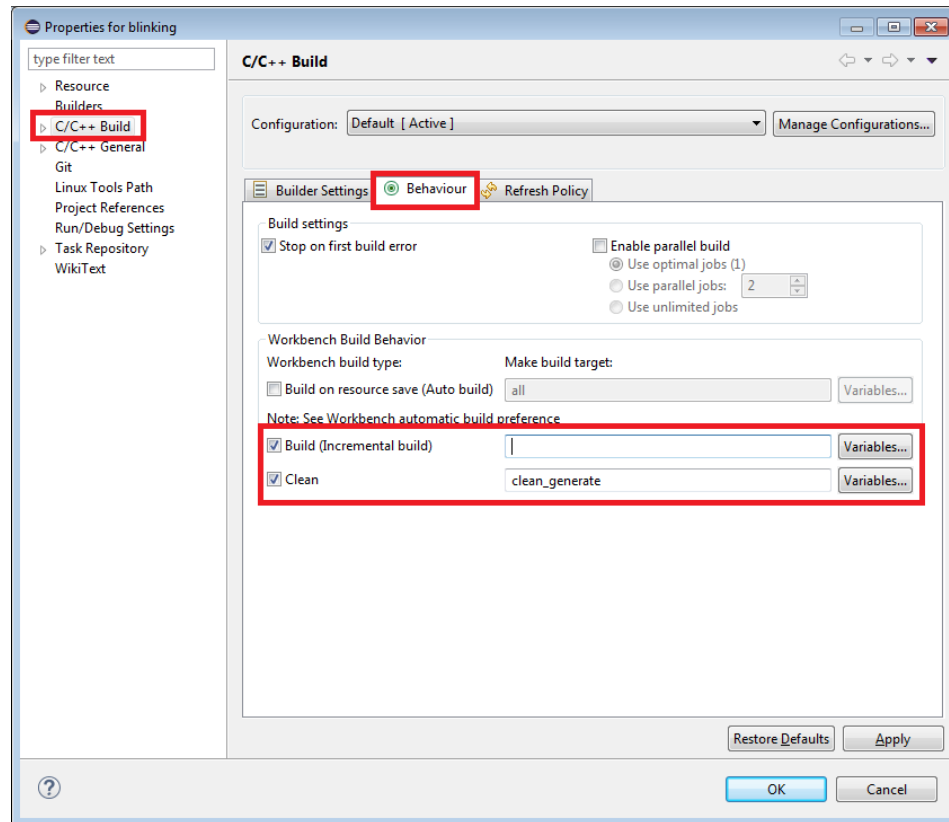


Figura 31: Configuración de comportamiento del IDE con el MakeFile

Luego de hacer esto, damos presionamos en **Ok**, y luego vamos sobre el proyecto Firmware, hacemos clic derecho, y buscamos la opción **Clean Project**. Por último, damos clic en **Build Project**. Dichos opciones se muestran en la Figura 32, y si todo salió correctamente, en la consola del IDE vamos a ver una ventana como la que se muestra en la Figura 33. Particularmente, vamos a ver una línea que nos dice que se ha creado un archivo Blinking.axf

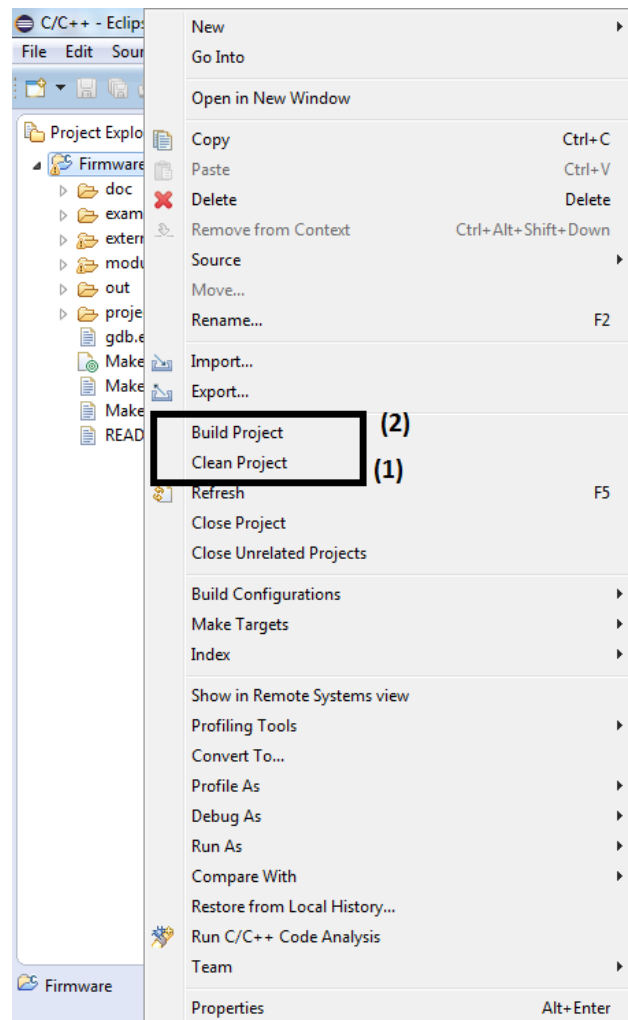


Figura 32: Menú de opciones del proyecto para hacer clean y build Project



Figura 33: Consola del software IDE luego de una compilación exitosa

6 Debug con Windows

Podríamos querer probar este ejemplo para ver de qué manera interactúa el software creado con el hardware de la EDU-CIAA, o verificar que no hay errores en nuestra lógica de programación. Para ello, debemos compilar el proyecto (Sección 5), y luego efectuar una depuración (*debug*). Aunque resulte trivial decirlo, si no contamos con el hardware, definitivamente no podremos hacer lo segundo. No obstante, para esos casos se ha desarrollado una pequeña aplicación para el ejemplo '*Blinking*' que hace funcionar la aplicación del Firmware en nuestra PC, generando las respuestas similares a las que haría la placa (si corriéramos el código en una EDU-CIAA, se produciría el encendido y apagado del LED). Aunque no sea tan lindo y emocionante como ver que las luces se prenden y se apagan, lo más cercano y simple que se puede hacer es que en cada conmutación del LED aparezca un mensaje en la consola del Software IDE que diga algo como '*Blinking*'. Esto es lo que haremos en la sección presente, y le daremos por nombre '*Win Debug*'. Dado que la aplicación compilada correrá en nuestra PC, el código está configurado para dar como resultado un archivo .exe.

Para que 'Win Debug' funcione debemos configurar un par de cosas. En principio debemos decirle al IDE dónde se encuentra el ejecutable que genera nuestro Makefile: haga click derecho sobre el proyecto, seleccione '*Debug As*' y a continuación '*Debug Configurations...*', como muestra la Figura 34.

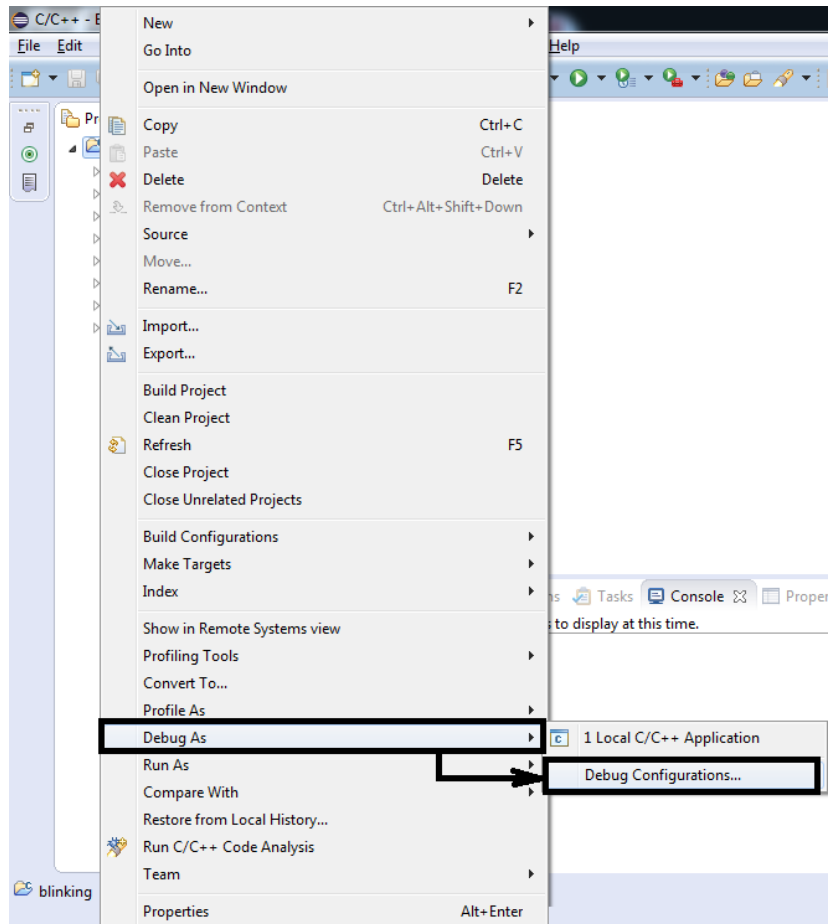


Figura 34: Acceso a Debug Configurations

Una vez hecho esto, hacemos doble click en la opción '**C/C++ Application**', lo cual nos creará un módulo donde podremos configurar un programa para que emule el comportamiento de la placa. El resultado se muestra en la Figura 35.

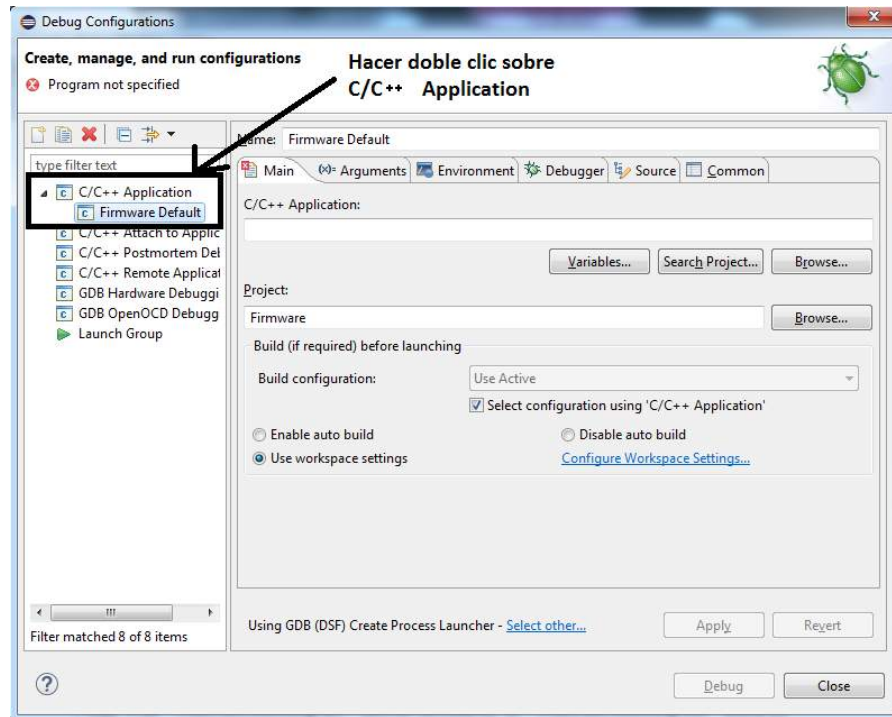


Figura 35: Creación de una opción de depuración por medio de un emulador

El ejecutable que genera el Makefile se almacena en la carpeta <Carpeta de instalación del software-IDE>\Firmware\out\bin\ bajo el nombre de **'blinking.exe'**. La configuración final se muestra en la Figura 36, y si se usan las rutas por defecto, los valores son los siguientes:

- C/C++ Application: **C:\CIAA\Firmware\out\bin\blinking.exe**
- Project: **blinking**
- Tildar la opción **'Use Workspace settings'**
- Tildar la opción **'Select configuration using 'C/C++ Application''**

Es muy posible que el IDE informe que no puede encontrar el Código Fuente cuando se intenta hacer Debug sobre este proyecto. Aparecerá una leyenda en el medio de la consola que se parece a la de la Figura 37. Para solucionarlo se debe crear un Path Mapping que convierta el estilo de ruta de CygWin al formato Windows. Abra la ventana de configuración del software IDE haciendo click en **'Windows→Preferences'**. Seleccione la rama **'C/C++'**, luego la rama **'Debug'** y ubique la opción **'Source Lookup Path'**. Finalmente, en el lado derecho haga click en el botón **'Add'**: aparecerá una ventana como la que se muestra en la Figura 38.

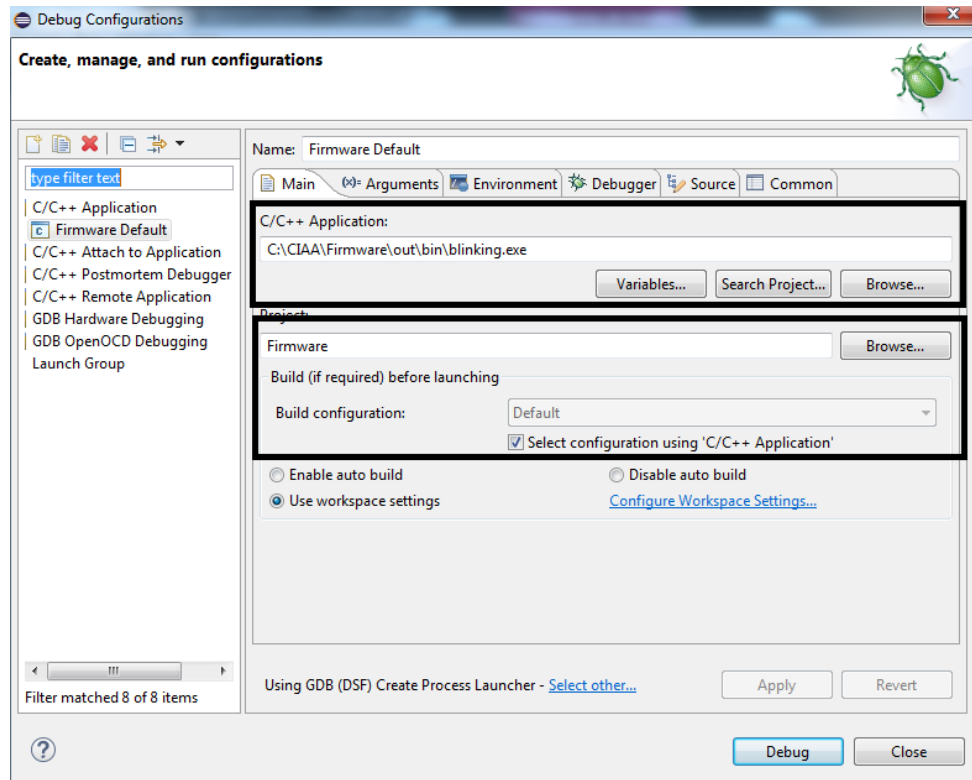


Figura 36: Configuración del emulador para el proyecto Blinking

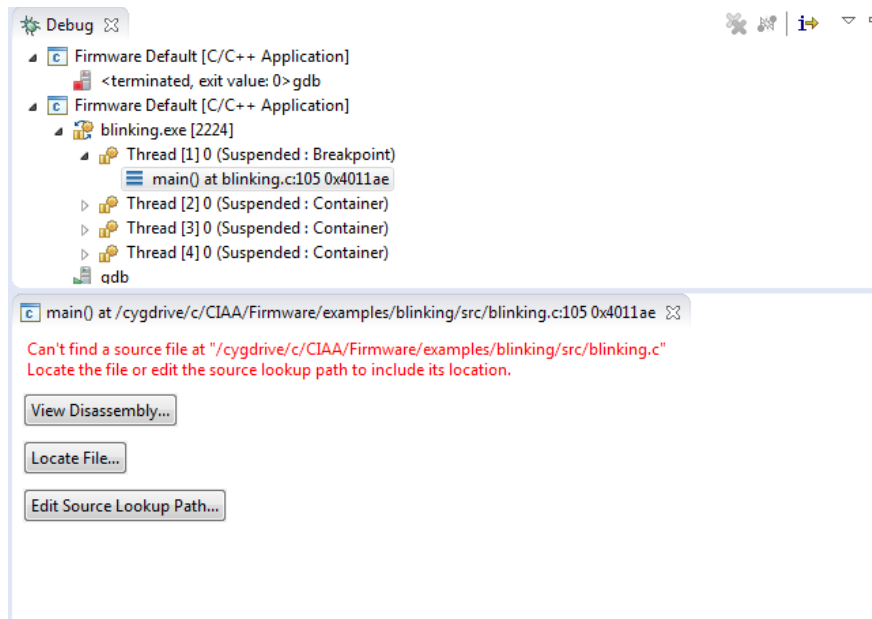


Figura 37: Error de IDE. No se encuentra el código fuente

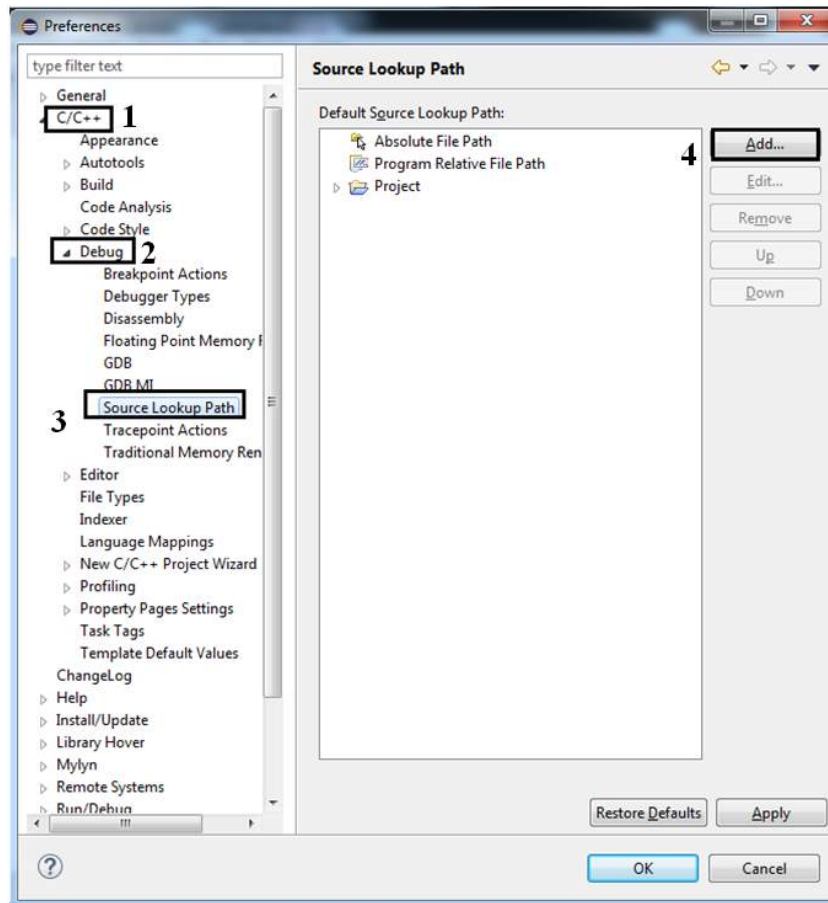


Figura 38: Configuración de mapeo de rutas para el entorno Windows

Cuando queramos agregar una fuente (*Source*), nos aparecerá el recuadro de la Figura 39, donde debemos elegir el tipo '*Path Mapping*'.

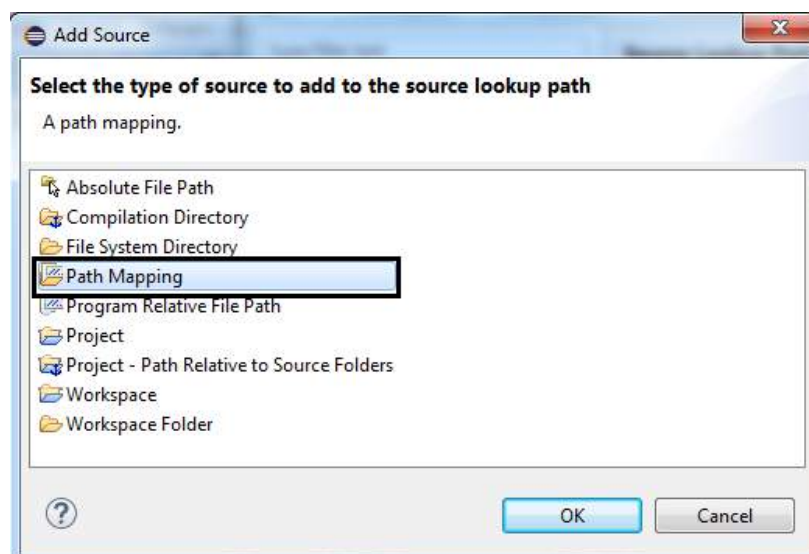


Figura 39: Agregando un Path Mapping

Nos aparecerá un nuevo recuadro, como el de la Figura 40, donde debemos elegir la opción '**Add**', y rellenamos con los siguientes datos:

- Name: ***Firmware***
- Compilation Path: ***\cygdrive\c\CIAA\Firmware***
- Local file system path: ***C:\CIAA\Firmware*** (si usamos los directorios por defecto)

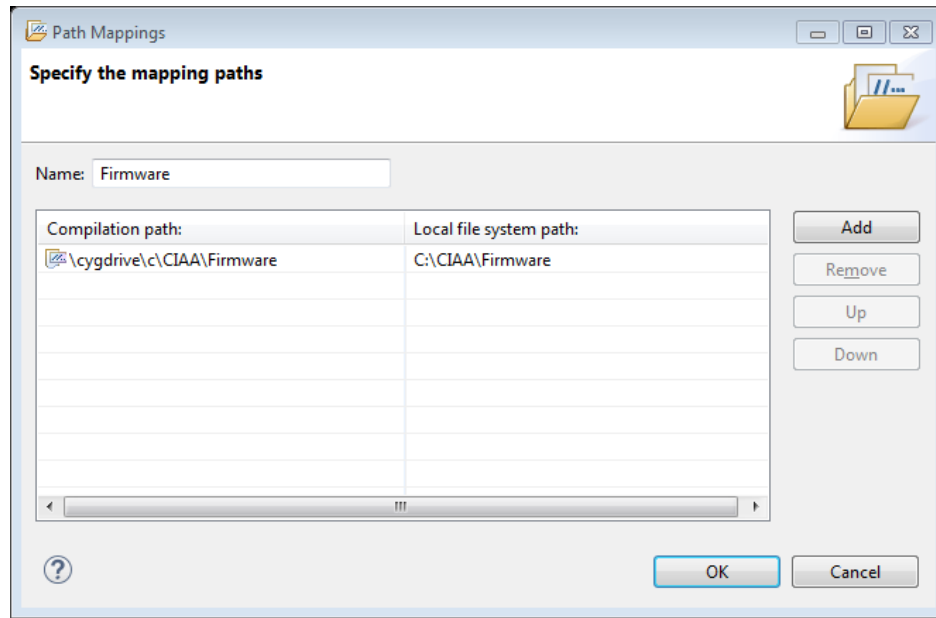


Figura 40: Configuración para crear un Path Mapping en Windows

Con todas estas configuraciones ya podremos depurar el ejemplo Blinking, emulando las respuestas a través de la consola.

Importante

Si bien la creación de un ***Path Mapping*** nos ayuda a emular el comportamiento del titileo de LEDs, nos genera conflictos cuando hacemos Debug con la propia placa, pues bloquea el uso de *breakpoints*. Si contamos con la placa y deseamos hacer Debug con la misma, debemos borrar el Path Mapping creado.

7 Debug en placa EDU-CIAA y Entorno IDE

7.1 Configuración del entorno CIAA-IDE

Si se cuenta con una placa EDU-CIAA y se quiere depurar sobre el hardware, lo primero que hay que hacer es una limpieza mediante '*Clean Project*'. Esto es necesario porque en todos los *builds* que se hicieron para el Win Debug se crearon archivos que pueden entorpecer la compilación del Debug sobre la placa, pues los mismos estaban pensados para el CPU de la PC, y no para el μC que forma parte de la EDU-CIAA.

A continuación necesitamos que el compilador reconozca que se quiere compilar sobre la placa EDU-CIAA. Ello implica que, en función de la placa disponible, se incluya el código acorde al μC utilizado. Dado que el código cuenta con sentencias *if* de pre-procesado, que dependen del hardware disponible, es necesario indicarle con qué versión de la plataforma se cuenta. Para ello, vamos a modificar manualmente el archivo MakeFile.mine, que es una rama del MakeFile, más reducido, que tiene los datos mínimos y necesarios para compilar. Para hacerlo, desde el software IDE, al mismo nivel que el proyecto Firmware (Figura 41), vamos a encontrar el MakeFile.mine que debemos modificar.

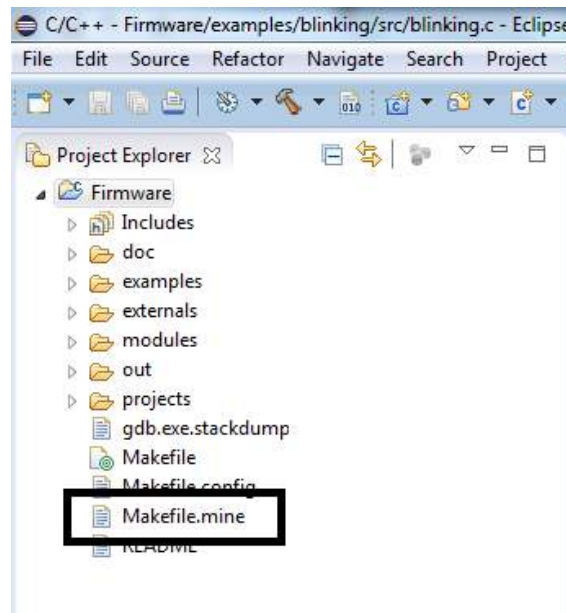


Figura 41: Project Explorer - MakeFile.mine

Le hacemos doble clic y nos va a aparecer a nuestra derecha, un archivo de texto. Si bajamos un poco, vamos a encontrar el código que se muestra en la Figura 42.


```
*Makefile.mine
52 #
53 #####
54 # ARCH, CPUTYPE and CPU following are supported
55 #
56 # | ARCH | CPUTYPE | CPU | COMPILER | BOARD |
57 # |-----|-----|-----|-----|-----|
58 # | x86 | ia32 | | gcc | ciao_sim_ia32 |
59 # | | ia64 | | gcc | ciao_sim_ia64 |
60 # |-----|-----|-----|-----|-----|
61 # | cortexM4 | lpc43xx | lpc4337 | gcc | edu_ciaa_nxp |
62 # | | k60_120 | mk60fx512vlq15 | gcc | ciao_nxp |
63 # | | | | gcc | ciao_fsl |
64 # |-----|-----|-----|-----|-----|
65 # | mips | pic32 | pic32mz | gcc | ciao_pic |
66 # |-----|-----|-----|-----|-----|
67 #
68 # if you define the BOARD the others parameters are optional.
69 #ARCH ?= x86
70 #CPUTYPE ?= ia32
71 #CPU ?= none
72 #COMPILER ?= gcc
73 #BOARD ?= ciao_sim_ia32
74 #####
75 # rtostests options
76 #
77 # RTOSTESTS_DEBUG_CTESTS ?= 1, set debug flag in ctest.pl
78 # RTOSTESTS_CLEAN_GENERATE ?= 1, skips make clean and generate (use it only if you are running a single sub test case, but after ger
79 # RTOSTESTS_CTEST ?= 'ctest_tm_01:Test Sequence 1', test case name, based on ctestcases.cfg (use it for single ctest)
80 # RTOSTESTS_SUBTEST ?= 'full-preemptive', sub test case name, based on ctestcases.cfg or empty to start running from this test to t
81
82 # Projects
83 #
```

Figura 42: Contenido del archivo MakeFile.mine

En él, podemos ver un cuadro donde, dependiendo de la plataforma en la que corramos el Firmware, debemos cambiar el valor de una variable que está debajo de él (la llamada BOARD. las otras no tiene efecto pues aparecen en forma de comentario, debido al símbolo # colocado al principio de cada línea). Como nosotros contamos con una EDU-CIAA, con micro NXP, entonces corresponde el siguiente valor de la variable:

BOARD ?= edu_ciaa_nxp

Y el archivo modificado nos quedará como se muestra en la Figura 43. Después de ello, tenemos que guardar los cambios.

```
Project Explorer
  Firmware
    Includes
    doc
    examples
    externals
    modules
    out
    projects
    gdb.exe.stackdump
    Makefile
    Makefile.config
    Makefile.mine
    README

*Makefile.mine
52 #
53 #####
54 # ARCH, CPUTYPE and CPU following are supported
55 #
56 # | ARCH | CPUTYPE | CPU | COMPILER | BOARD |
57 # |-----|-----|-----|-----|-----|
58 # | x86 | ia32 | | gcc | ciao_sim_ia32 |
59 # | | ia64 | | gcc | ciao_sim_ia64 |
60 # |-----|-----|-----|-----|-----|
61 # | cortexM4 | lpc43xx | lpc4337 | gcc | edu_ciaa_nxp |
62 # | | k60_120 | mk60fx512vlq15 | gcc | ciao_nxp |
63 # | | | | gcc | ciao_fsl |
64 # |-----|-----|-----|-----|-----|
65 # | mips | pic32 | pic32mz | gcc | ciao_pic |
66 # |-----|-----|-----|-----|-----|
67 #
68 # if you define the BOARD the others parameters are optional.
69 #ARCH ?= x86
70 #CPUTYPE ?= ia32
71 #CPU ?= none
72 #COMPILER ?= gcc
73 #BOARD ?= edu_ciaa_nxp
74 #####
75 # rtostests options
76 #
77 # RTOSTESTS_DEBUG_CTESTS ?= 1, set debug flag in ctest.pl
78 # RTOSTESTS_CLEAN_GENERATE ?= 1, skips make clean and generate (use it only if you are running a single sub test case, but after ger
79 # RTOSTESTS_CTEST ?= 'ctest_tm_01:Test Sequence 1', test case name, based on ctestcases.cfg (use it for single ctest)
80 # RTOSTESTS_SUBTEST ?= 'full-preemptive', sub test case name, based on ctestcases.cfg or empty to start running from this test to t
81
82 # Projects
83 #
84 # Available projects are:
```

Figura 43: MakeFile.mine luego de la configuración

7.2 Compilación del proyecto

Una vez configurado el MakeFile, se debe limpiar (**Clean Project**) y luego compilar el proyecto (**Build Project**). Como se puede ver en la Figura 44, la extensión del archivo generado ya no es **.exe** sino **.axf**. Esta extensión es propia de la arquitectura ARM. Si esto no ocurre, vuelva al paso anterior, y verifique que no existen errores de tipeo en el nombre o en el valor de las variables que se modificaron en los MakeFile.

Para saber si se produjo una correcta compilación, vea la consola del IDE, ubicada en la parte inferior de la pantalla. En caso de que se hayan seguido todos los pasos y no se pueda compilar, hacer un **Clean** primero y luego un **Build**.

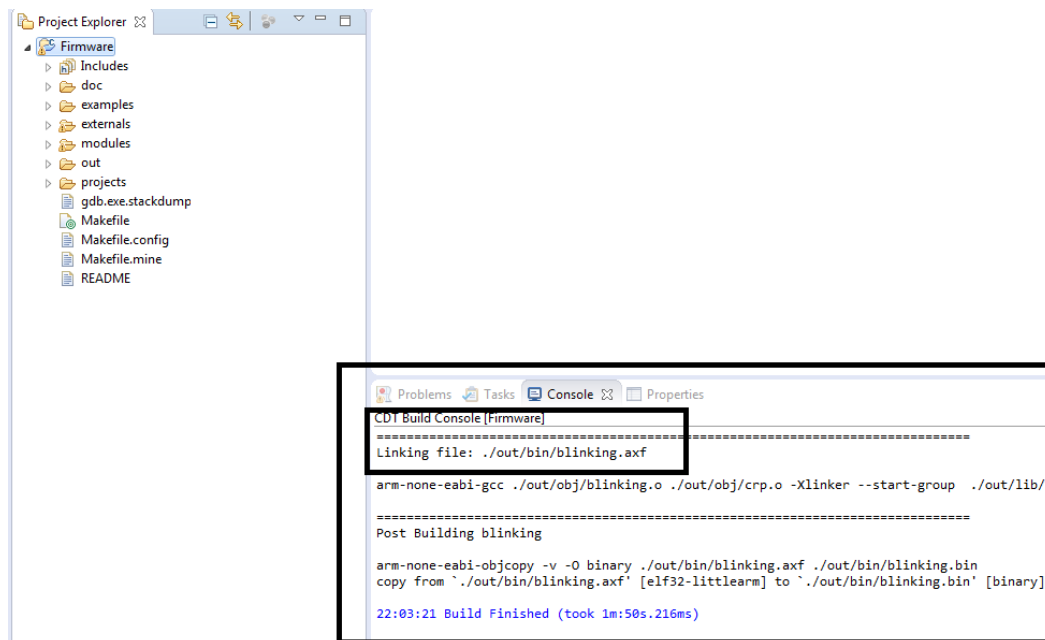


Figura 44: Entorno del Software-IDE luego de la primera compilación

7.3 Depuración sobre la placa: configuración de OpenOCD para Debug

Durante la instalación del CIAA-IDE, además de los drivers para la conexión de la placa se instalan las herramientas para Debug. Se utilizará OpenOCD (una herramienta OpenSource creada para estos propósitos) para hacer el nexo JTAG-GDB mediante la conexión USB. Para configurar esta herramienta, desde el menú '**Run→Debug Configurations...**' se debe crear un módulo nuevo de '**Debug configuration**' del tipo '**GDB OpenOCD Debugging**'. A continuación,

coloque los valores que se muestran en las Figura 45 y Figura 46. Las mismas se listan a continuación:

- Pestaña Main:
 1. Name: ***Blinking OpenOCD***
 2. Project: ***blinking***
 3. C/C++ Application: ***C:\CIAA\Firmware\out\bin\blinking.axf***
 4. Tildar ***Disable auto build***
 5. Build configuration: ***Default***
- Pestaña Debugger:
 1. OpenOCD Setup: Destildar la opción ***Start OpenOCD locally***
 2. GDB Client Setup
 - a) Executable: ***C:\CIAA\cygwin\usr\arm-none-eabi\bin\arm-none-eabi-gdb.exe***
 - b) Other options y Commands: dejarlo como está
 - c) Destildar ***'Force thread list update on suspend'***

Le damos clic a ***Apply*** para guardar los cambios, y no cerramos esta ventana.

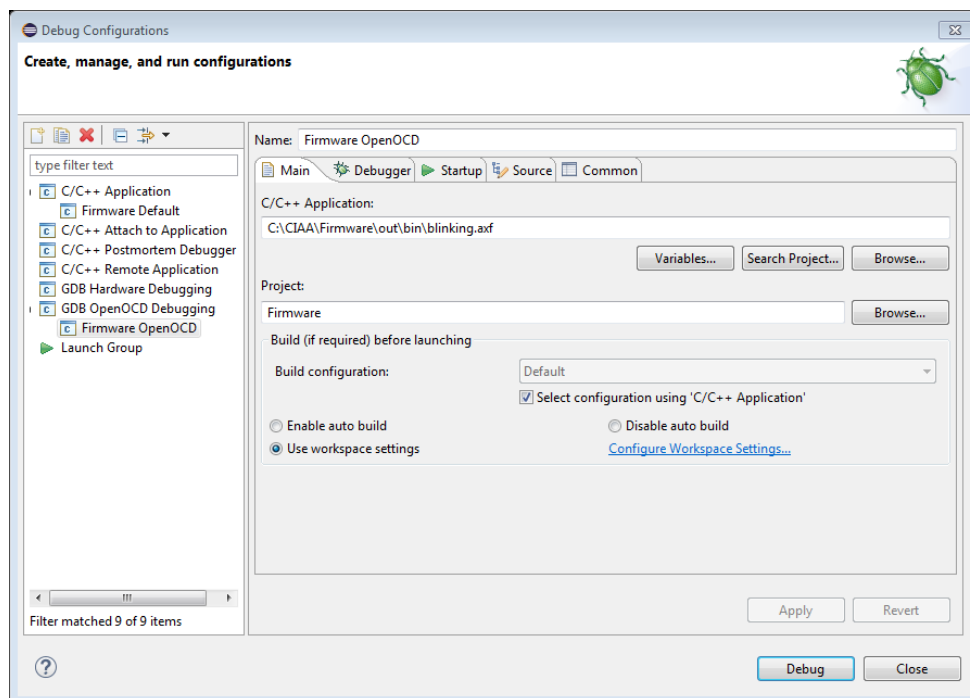


Figura 45: Configuración para poder hacer Debug sobre la placa: pestaña Main

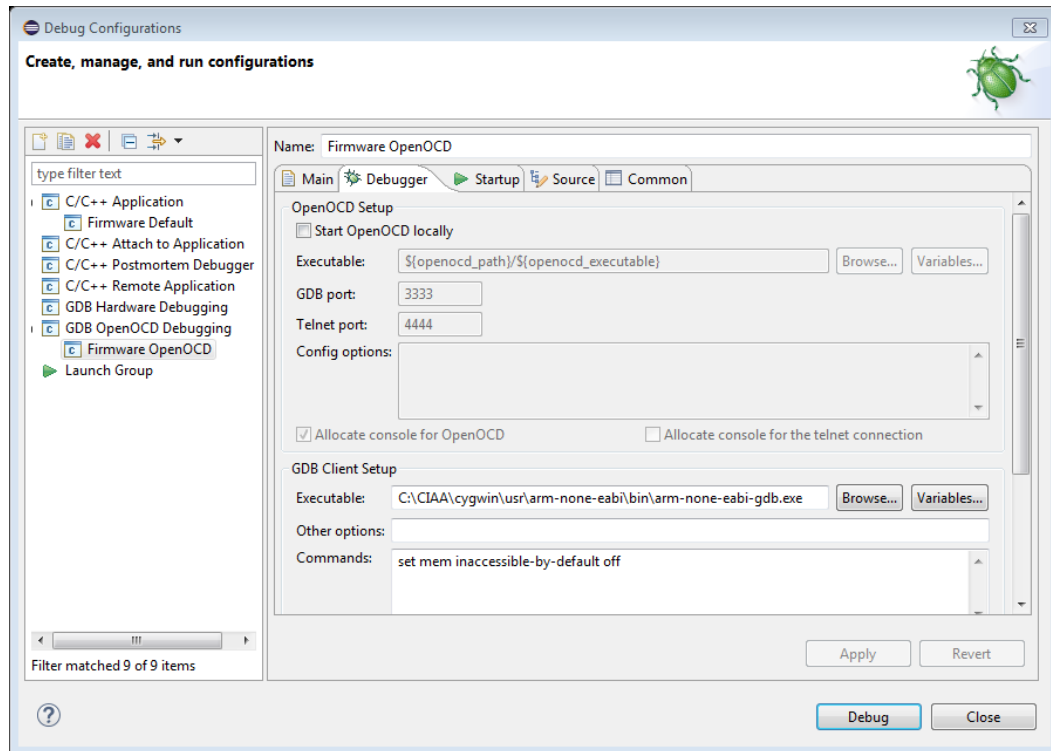


Figura 46: Configuración para poder hacer Debug sobre la placa: pestaña Debugger

Para hacer Debug, es necesario que se abra un puerto a través del OpenOCD. Hay muchas formas de hacerlo. En este tutorial, lo haremos usando la consola CygWin, que viene incluida con el instalador del CIAA-IDE.

Abrimos la consola. Nos aparecerá una ventana como la que se muestra en la Figura 47.

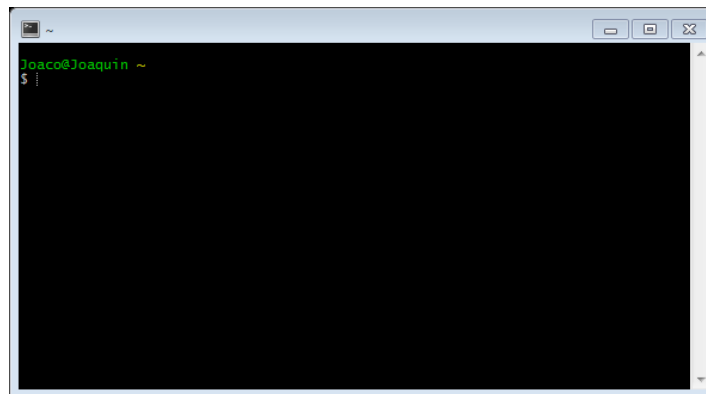


Figura 47: Consola CygWin

Entonces, escribimos los siguientes comandos:

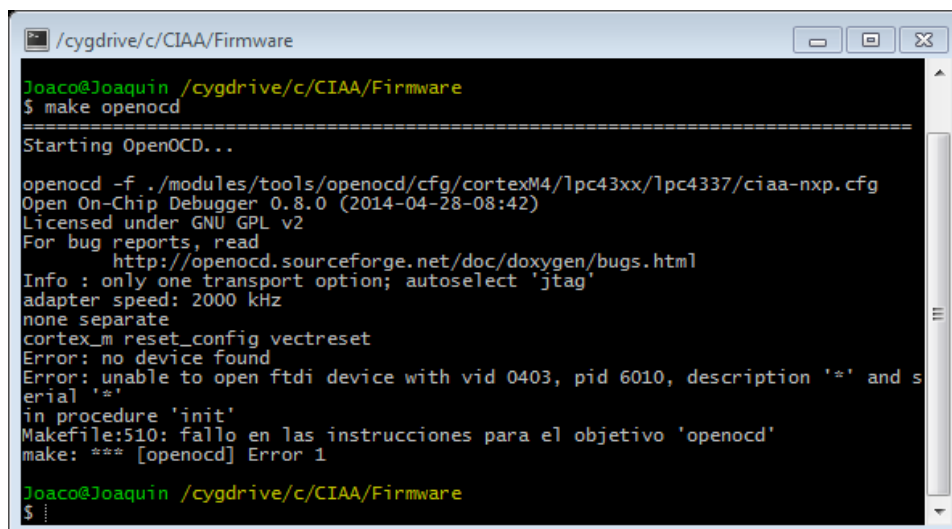
Comando	Descripción
cd C:/CIAA/Firmware	Nos posiciona en la carpeta Firmware (si usamos los Path por defecto)
make openocd	Comienza a correr el servicio del OpenOCD

Si se cuenta con una Placa EDU-CIAA “virgen”, cuya flash nunca ha sido programada, o si por algún motivo se ha borrado completamente la flash del microcontrolador, es posible que al intentar iniciar una sesión de Debug, el CIAA-IDE muestre un error. Para poder iniciar la sesión de debug, recomendamos seguir las instrucciones de la sección **Primeros pasos con el Hardware de la CIAA**, accesible en la página oficial del proyecto.

Luego de iniciado el servicio del OpenOCD, dejamos la ventana abierta, volvemos al entorno IDE, y damos clic al botón **Debug** de la ventana que dejamos abierta previamente.

7.4 Posible problema: “No reconocimiento”

Si se alcanzó este punto, si todo fue exitoso, al intentar hacer debug sobre la placa no debería haber problemas. No obstante, si por casualidad se desconectó la placa de la PC y se la volvió a conectar en un puerto USB distinto al anterior, Windows detectará el nuevo Hardware encontrado, instalará los drivers y la placa se encenderá; pero al intentar hacer Debug, en la consola CygWin es posible que emerja un mensaje de error como la de la Figura 48.



```
/cygdrive/c/CIAA/Firmware
Joaco@Joquin /cygdrive/c/CIAA/Firmware
$ make openocd
=====
Starting OpenOCD...
openocd -f ./modules/tools/openocd/cfg/cortexM4/lpc43xx/lpc4337/ciaa-nxp.cfg
Open On-Chip Debugger 0.8.0 (2014-04-28-08:42)
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.sourceforge.net/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'jtag'
adapter speed: 2000 kHz
none separate
cortex_m reset_config vectreset
Error: no device found
Error: unable to open ftdi device with vid 0403, pid 6010, description '*' and s
erial '*'
in procedure 'init'
Makefile:510: fallo en las instrucciones para el objetivo 'openocd'
make: *** [openocd] Error 1

Joaco@Joquin /cygdrive/c/CIAA/Firmware
$
```

Figura 48: Ventana de error del Software IDE

Si se hiciera un par de pruebas, este error también corresponde al que se daría si la placa no estuviera conectada. La causa es que el driver que se acaba de instalar no es el corregido por el programa Zadig, sino es el instalado por el instalador del Software-IDE. Para solucionar este inconveniente simplemente hay que repetir los pasos que se destacan en la **Sección 3.5**, y con ello, la placa vuelve a funcionar correctamente.